

Отчёт по лабораторной работе 9

Архитектура компьютеров

Ермишина Мария Кирилловна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программ с помощью GDB	9
2.3	Задание для самостоятельной работы	21
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 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

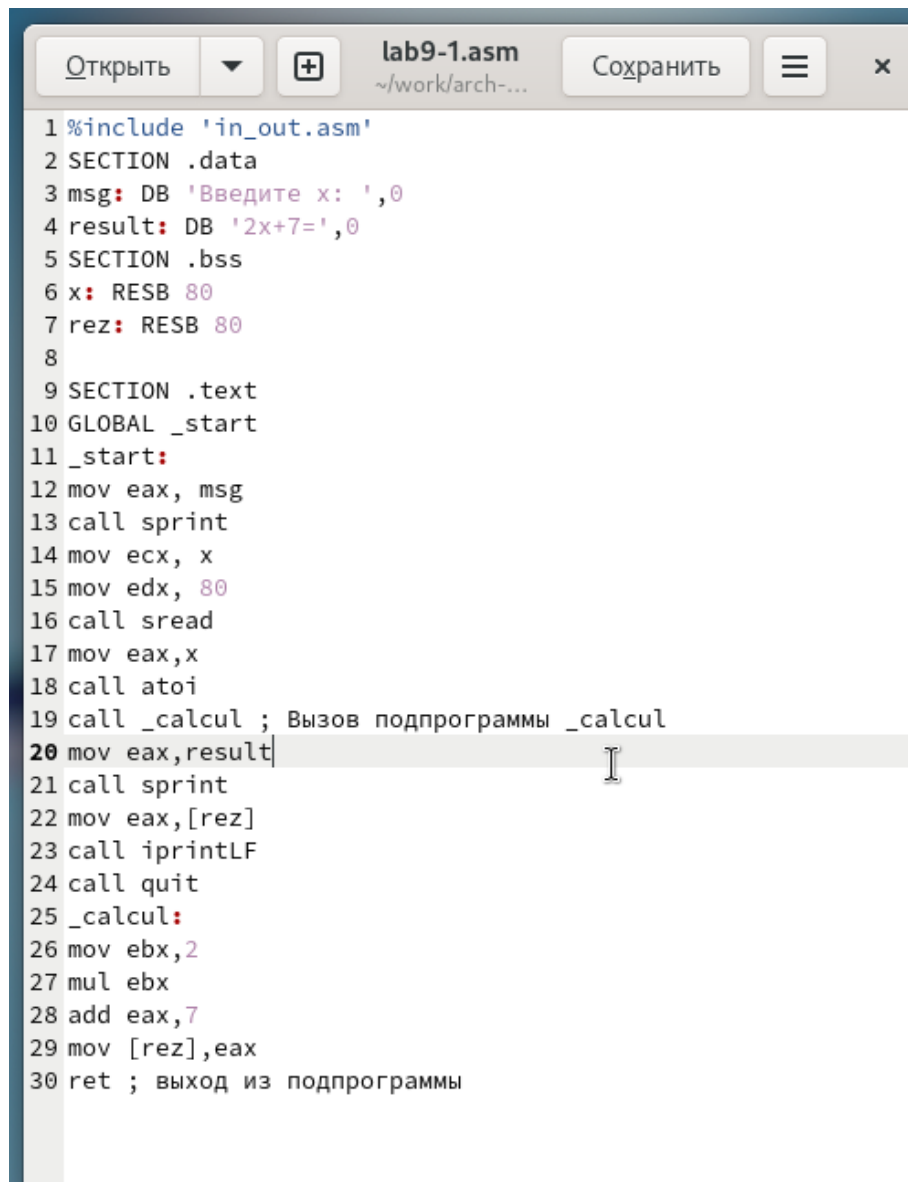
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

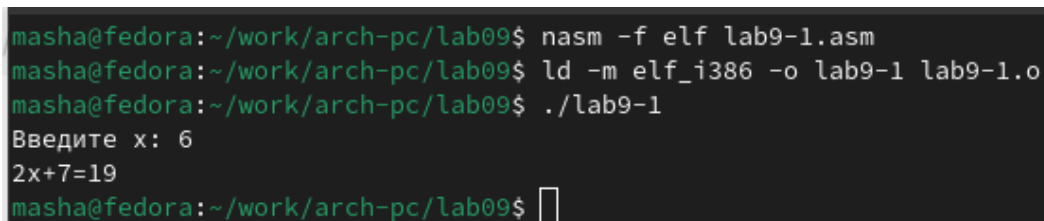
Сначала я создала новую папку, чтобы выполнять лабораторную работу номер 9, и перешла в нее. Затем я создала файл с именем lab9-1.asm.

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)



```
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
```

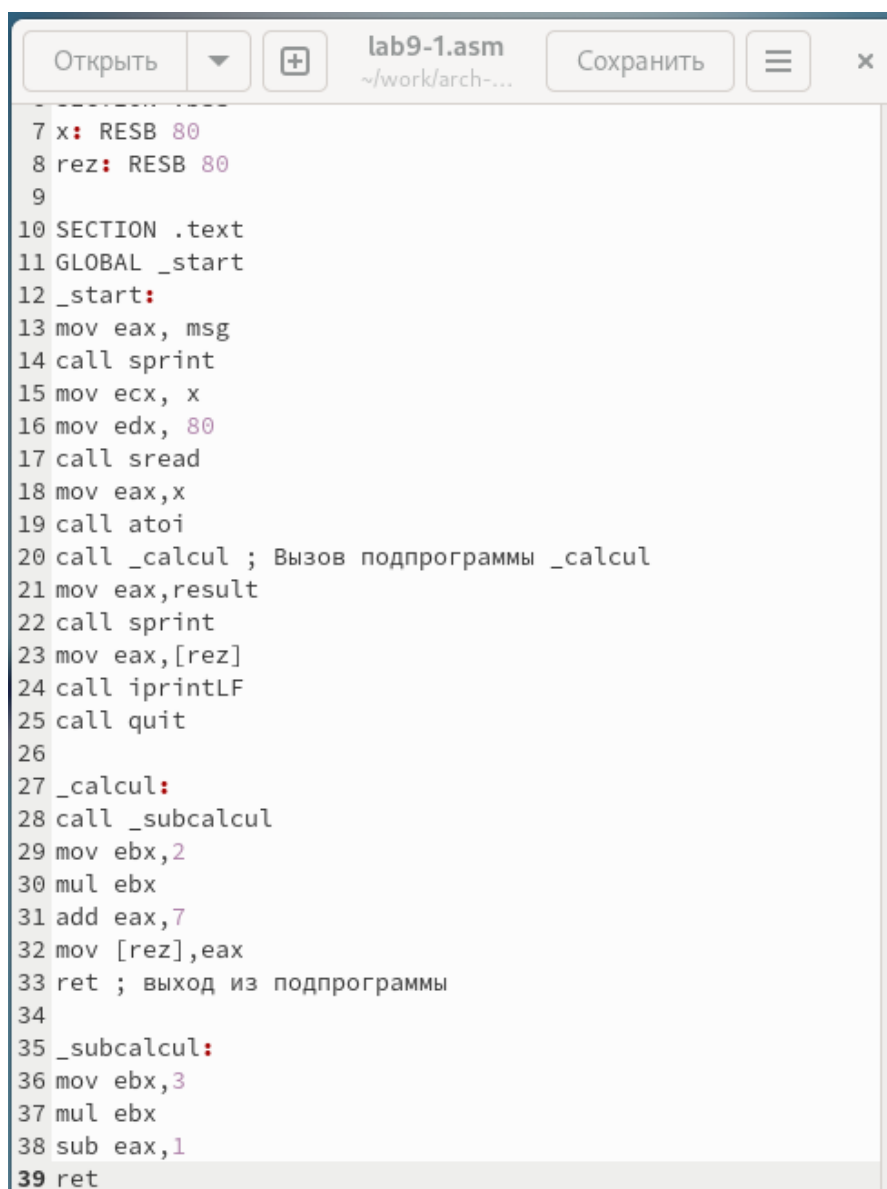
Рис. 2.1: Программа lab9-1.asm



```
masha@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
masha@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
masha@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2x+7=19
masha@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)



```
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
```

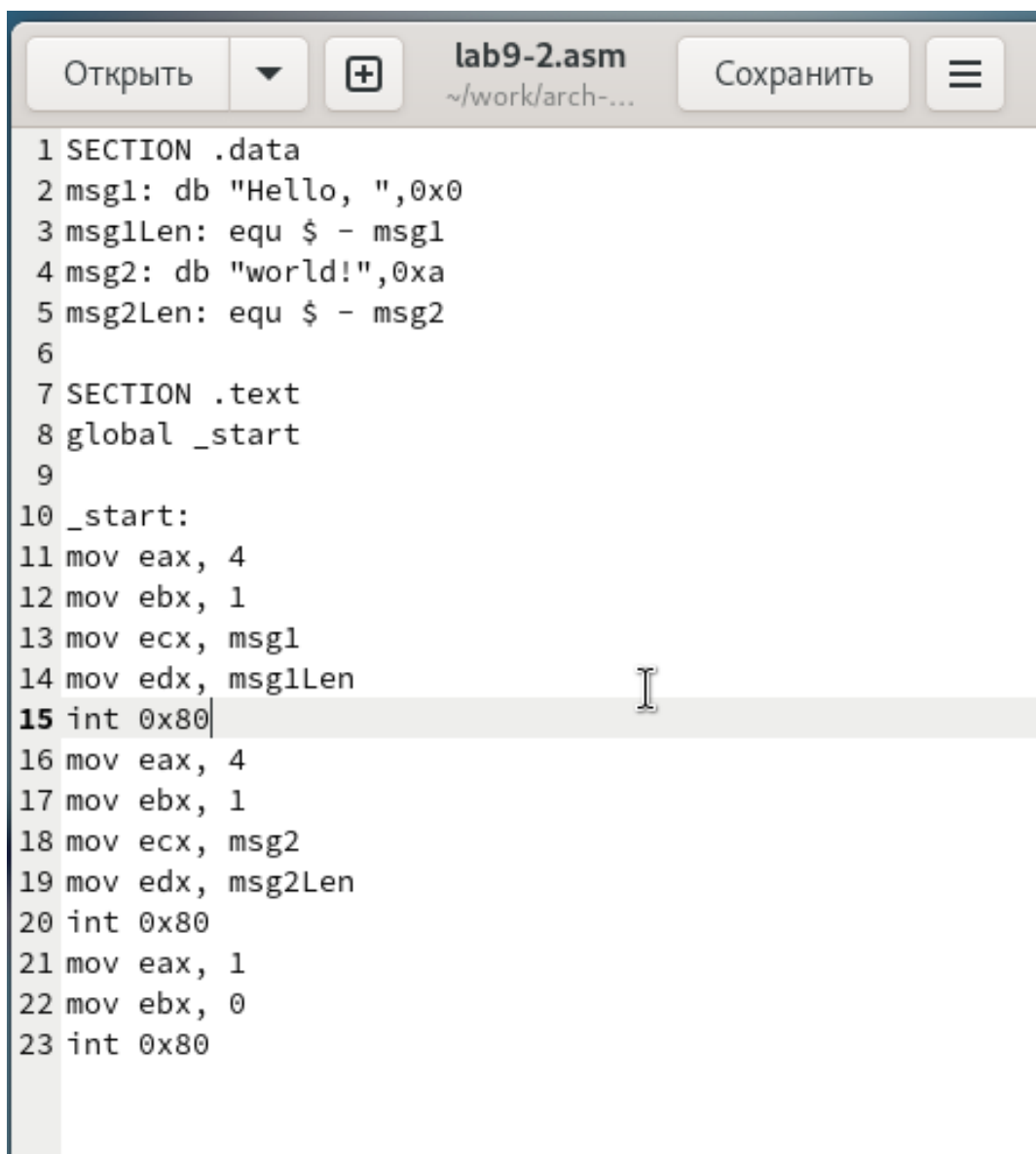
Рис. 2.3: Программа lab9-1.asm


```
masha@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
masha@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
masha@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2x+7=19
masha@fedora:~/work/arch-pc/lab09$
masha@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
masha@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
masha@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2(3x-1)+7=41
masha@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)



```
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

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)

```

masha@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
masha@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
masha@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/masha/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 3559) 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)

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

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/masha/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 3559) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/masha/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

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

Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/masha/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) 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)

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

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35

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
0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0

native process 3564 (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 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)

```
masha@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 ]
cs       0x23     35

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
0x804902c <_start+44>   mov     eax,0x1
b+ 0x8049031 <_start+49> mov     ebx,0x0

native process 3564 (asm) In: _start L12 PC: 0x8049005
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb)
```

Рис. 2.10: Изменение регистров

```
masha@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 ]
cs       0x23     35

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
0x804902c <_start+44>   mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0

native process 3564 (asm) In: _start L16 PC: 0x8049016
cs      0x23     35
ss      0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
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)


```
masha@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 ]
cs       0x23     35

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
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>     mov     ebx,0x0

native process 3564 (asm) In: _start      L16      PC: 0x8049016
(gdb) si
(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

```
masha@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 ]
cs       0x23     35

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
   0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0

native process 3564 (asm) In: _start L16 PC: 0x8049016
(gdb) p/t $eax
$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 titled "masha@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 various CPU registers:

Register	Value	Comment
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]
cs	0x23	35

. 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
0x804902a <_start+42>     int      0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>    mov     ebx,0x0
```

. The bottom section shows the command prompt "(gdb)" and a series of commands and their outputs:

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

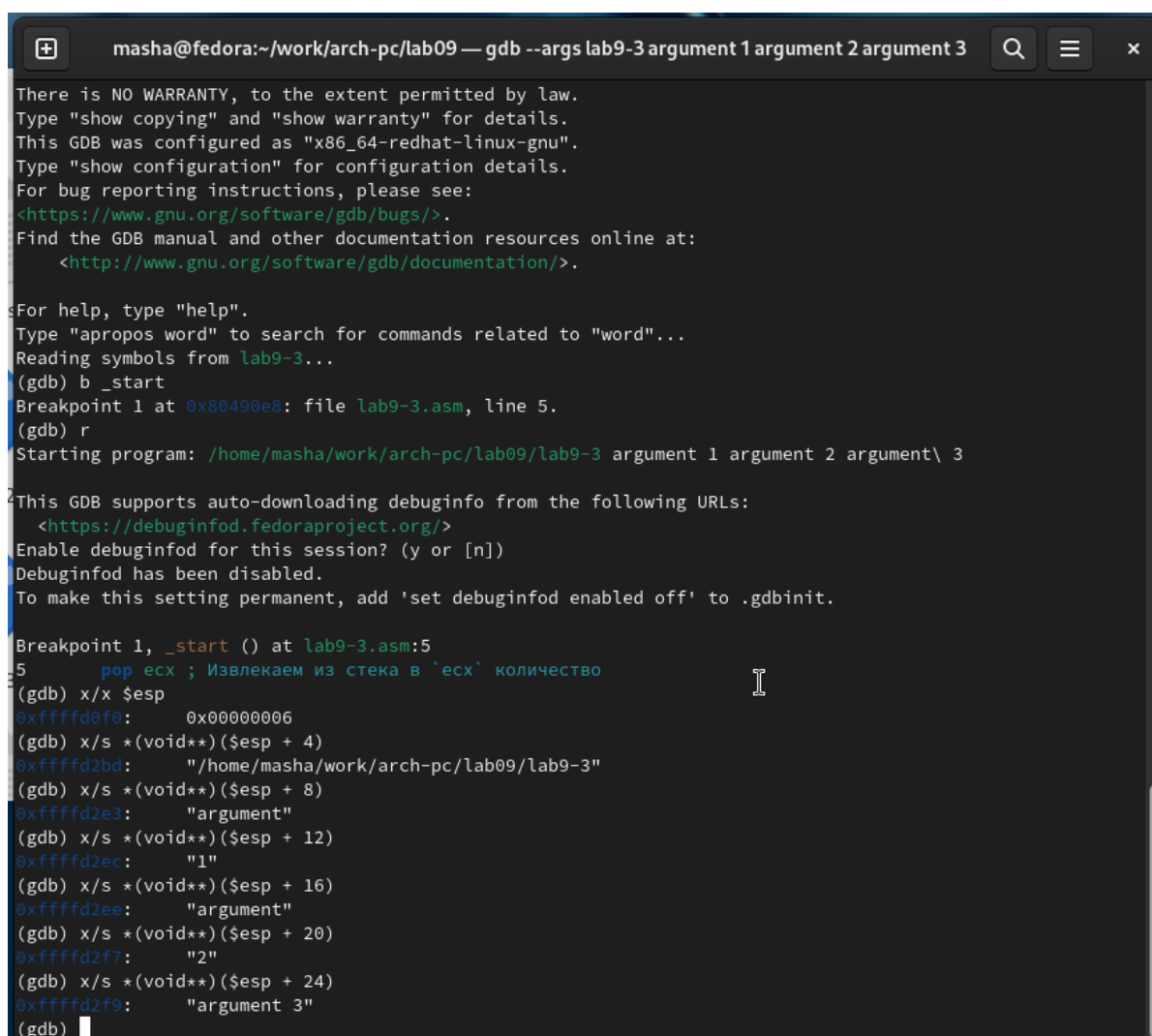
Я скопировала файл lab8-2.asm, который был создан во время выполнения лабораторной работы №8. Этот файл содержит программу для вывода аргументов командной строки. Затем я создала исполняемый файл из скопированного файла.

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)



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

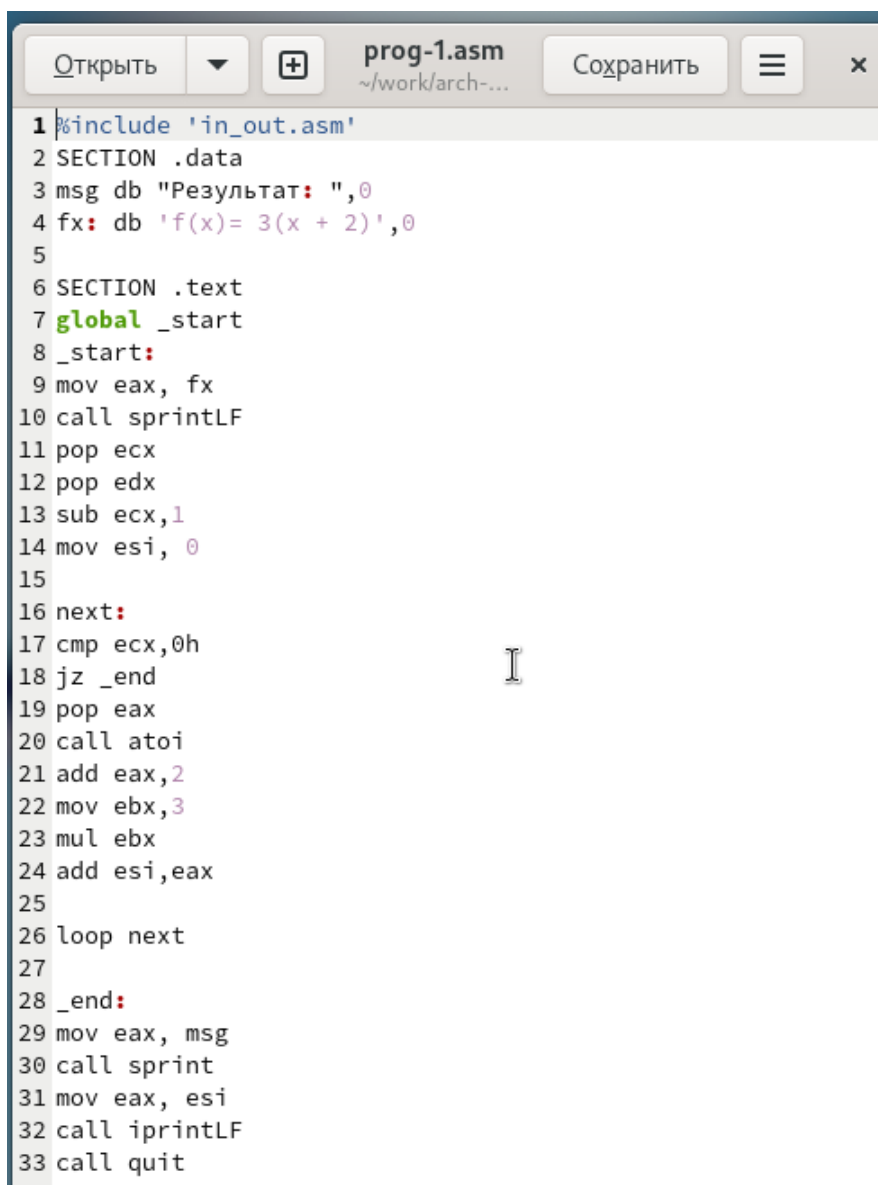
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

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)
0xffffd2bd: "/home/masha/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2e3: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2ec: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2ee: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2f7: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2f9: "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)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 3(x + 2)',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
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 add eax,2
22 mov ebx,3
23 mul ebx
24 add esi,eax
25
26 loop next
27
28 _end:
29 mov eax, msg
30 call sprint
31 mov eax, esi
32 call iprintLF
33 call quit
```

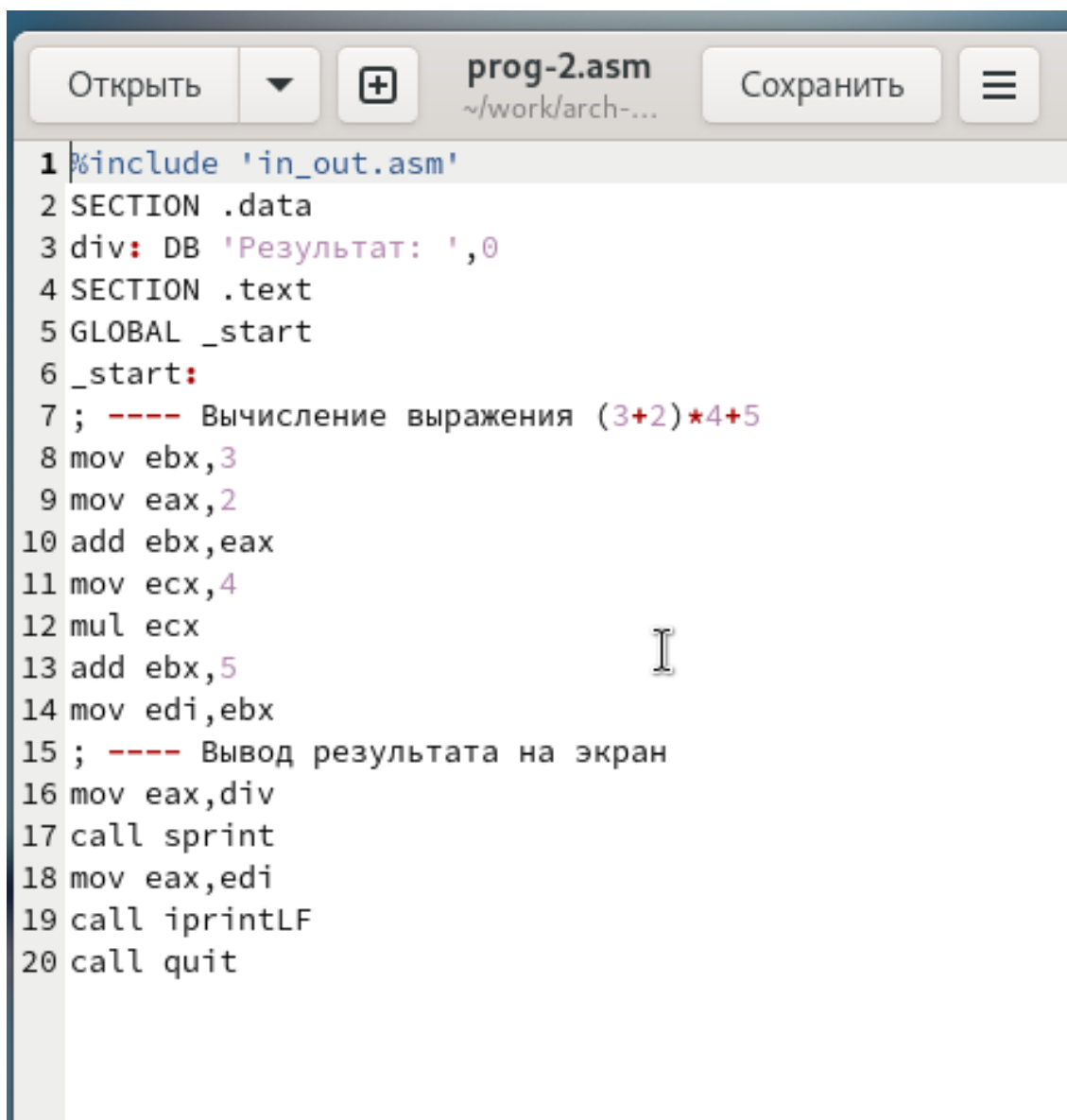
Рис. 2.16: Программа prog-1.asm

```
masha@fedora:~/work/arch-pc/lab09$  
masha@fedora:~/work/arch-pc/lab09$ nasm -f elf prog-1.asm  
masha@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 prog-1.o -o prog-1  
masha@fedora:~/work/arch-pc/lab09$ ./prog-1  
f(x)= 3(x + 2)  
Результат: 0  
masha@fedora:~/work/arch-pc/lab09$ ./prog-1 3  
f(x)= 3(x + 2)  
Результат: 15  
masha@fedora:~/work/arch-pc/lab09$ ./prog-1 6 4 1 3 7  
f(x)= 3(x + 2)  
Результат: 93  
masha@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)



The screenshot shows a code editor window titled "prog-2.asm" with a path "~/work/arch-...". The editor contains 20 lines of assembly code. A cursor is positioned at the end of line 12, which is "mul ecx". The code is as follows:

```
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

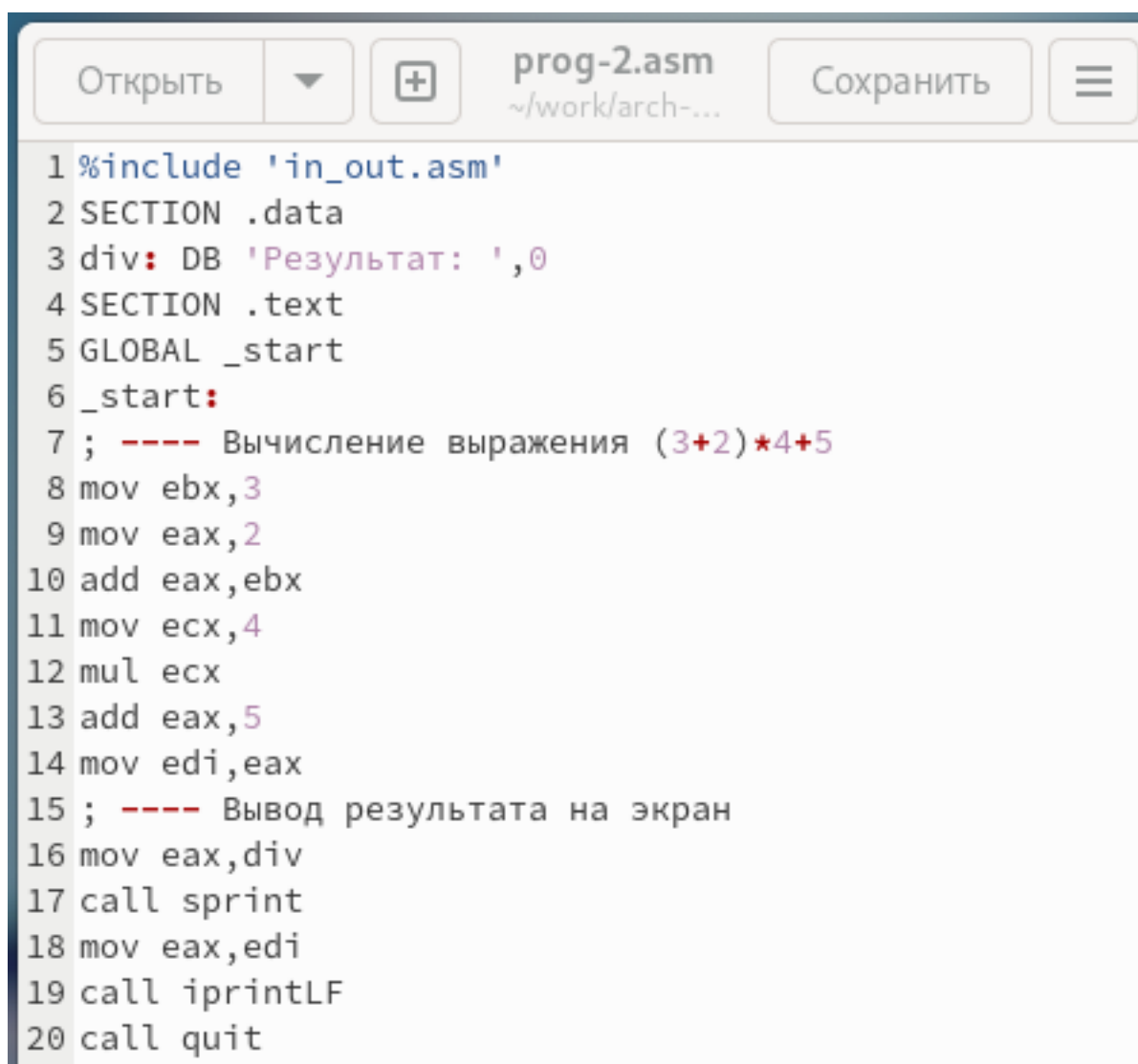
The screenshot shows a GDB window titled "masha@fedora:~/work/arch-pc/lab09 — gdb prog-2". The top panel displays register values, with "eax" set to "s" and "8". Below this, a message "[Register Values Unavailable]" is shown. The main panel displays assembly code starting at address 0x80490fe, which is labeled as "<_start+22>". The code includes instructions for moving data from ebx to edi, calling sprint, moving edi to eax, calling iprintLF, and calling quit. The bottom panel shows the GDB prompt with the command "si" being entered multiple times, followed by "c" to continue execution. The output indicates that the process exited normally with a result of 10.

```
masha@fedora:~/work/arch-pc/lab09 — gdb prog-2
eax      s      8
[ Register Values Unavailable ]
0x80490fe 0490fe x80490fe <_start+22>
0x80490fe <_start+22> mov edi,ebx04a000
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 3815 (asm) In: _start L14 PC: 0x80490fe
DebuginNo process (asm) In: L?? PC: ??
Breakpoint 1, _start () at prog-2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 3815) exited normally]
(gdb)
```

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)



The image shows a screenshot of an assembly code editor window. The window has a title bar with the filename "prog-2.asm" and a path "~/work/arch-...". There are buttons for "Открыть" (Open), "Сохранить" (Save), and a menu icon. The code is as follows:

```
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

```
masha@fedora:~/work/arch-pc/lab09 — gdb prog-2
eax 5 25

fffd120 xffffd120
[ Register Values Unavailable ]

0490fe x80490fe <_start+22>

0x80490fe <_start+22> mov edi,eax04a000
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 3909 (asm) In: _start L14 PC: 0x80490fe
Debug: No process (asm) In: L?? PC: ??

Breakpoint 1, _start () at prog-2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 3909) exited normally]
(gdb)
```

Рис. 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.