

# **Отчёт по лабораторной работе 9**

**Архитектура компьютеров**

**Гамбердов С.И. НПИбд-03-24**

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	25

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

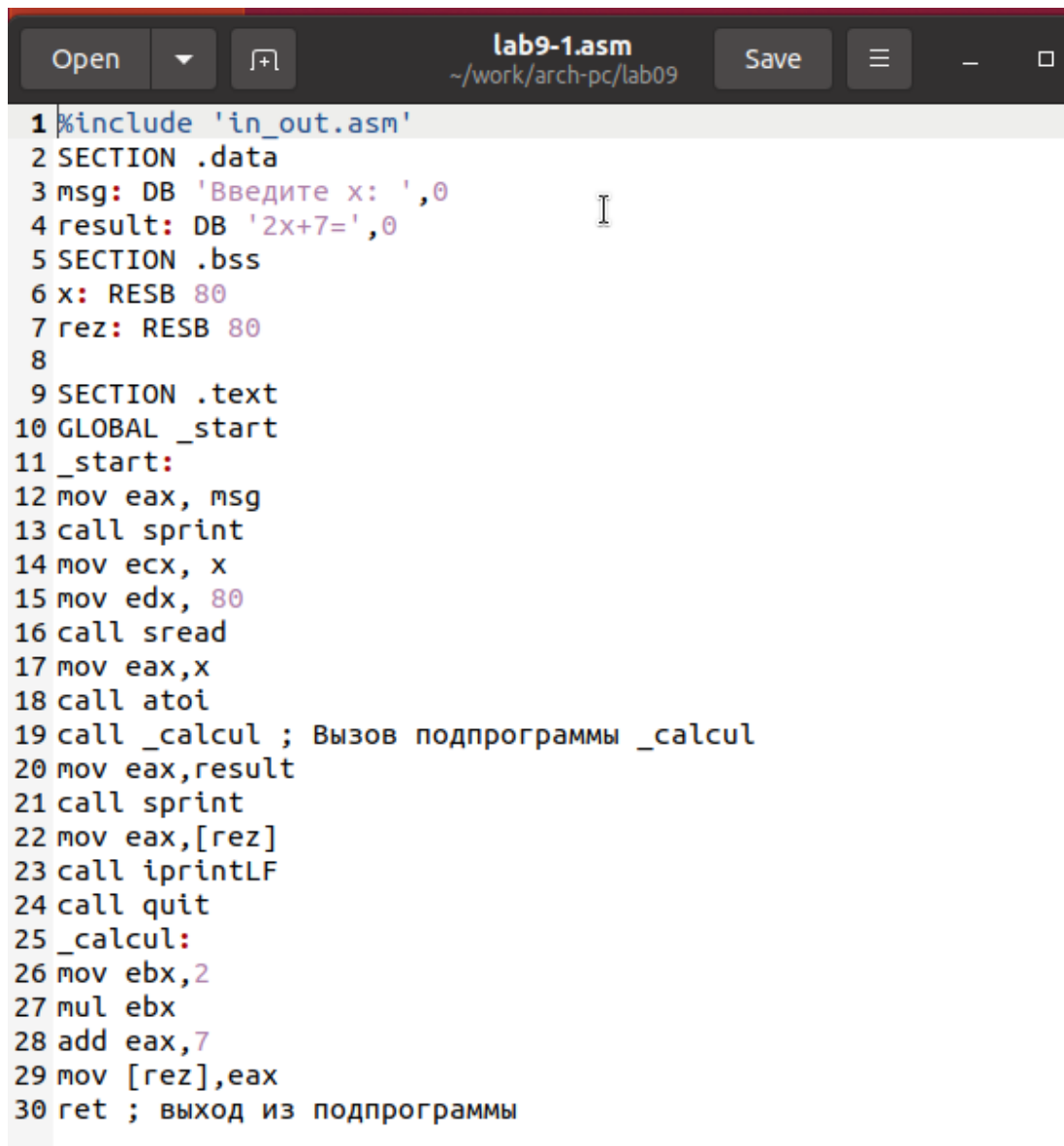
## Список таблиц

# 1 Цель работы

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

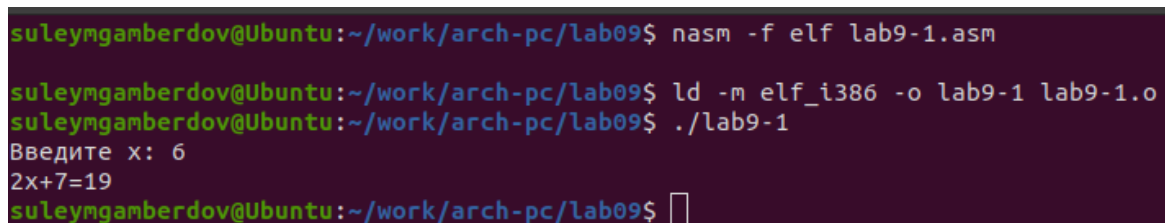
## 2 Выполнение лабораторной работы

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
```

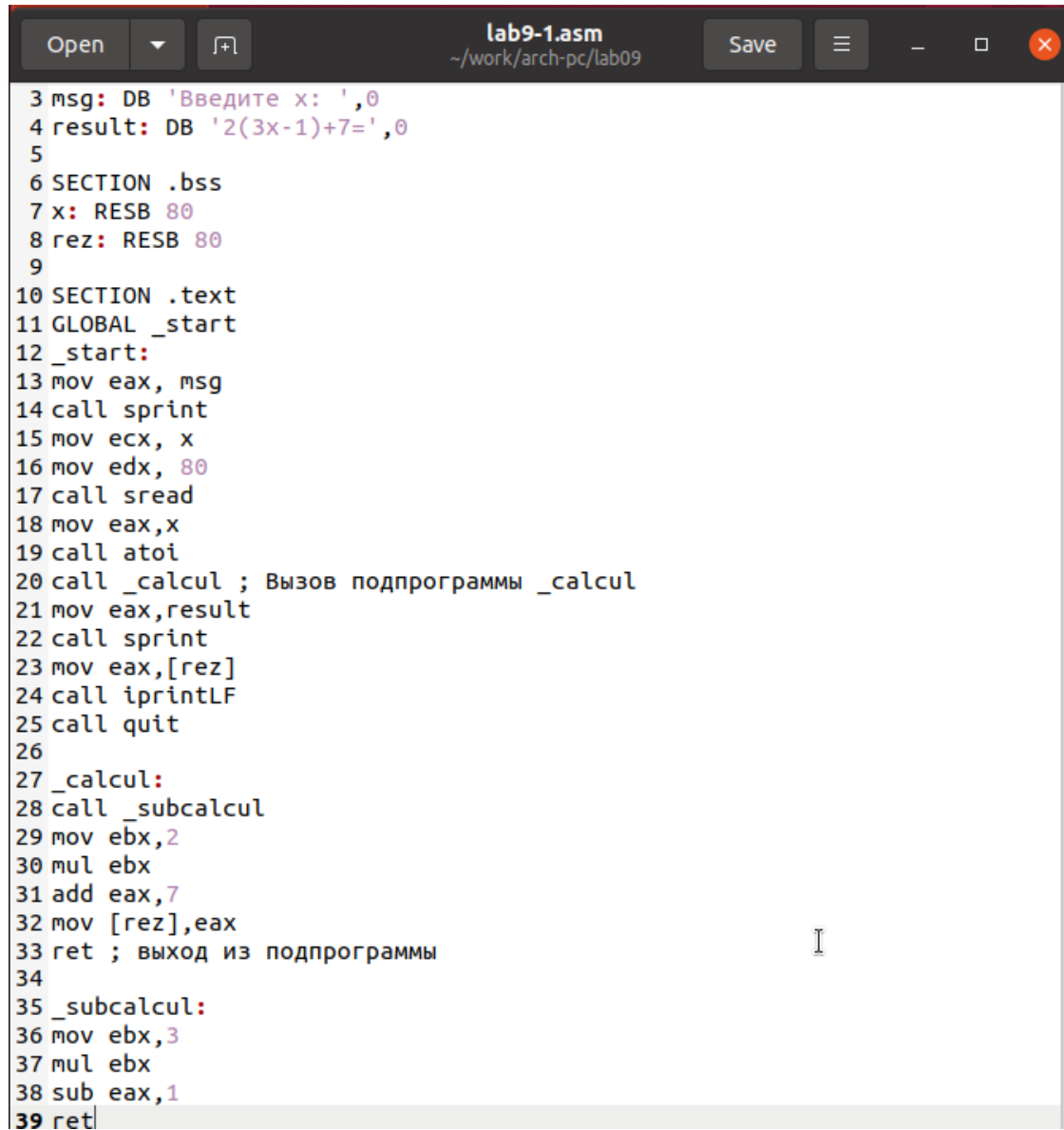
Рис. 2.1: Программа lab9-1.asm



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

Рис. 2.2: Запуск программы lab9-1.asm

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



```
3 msg: DB 'Введите x: ',0
4 result: DB '2(3x-1)+7=',0
5
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



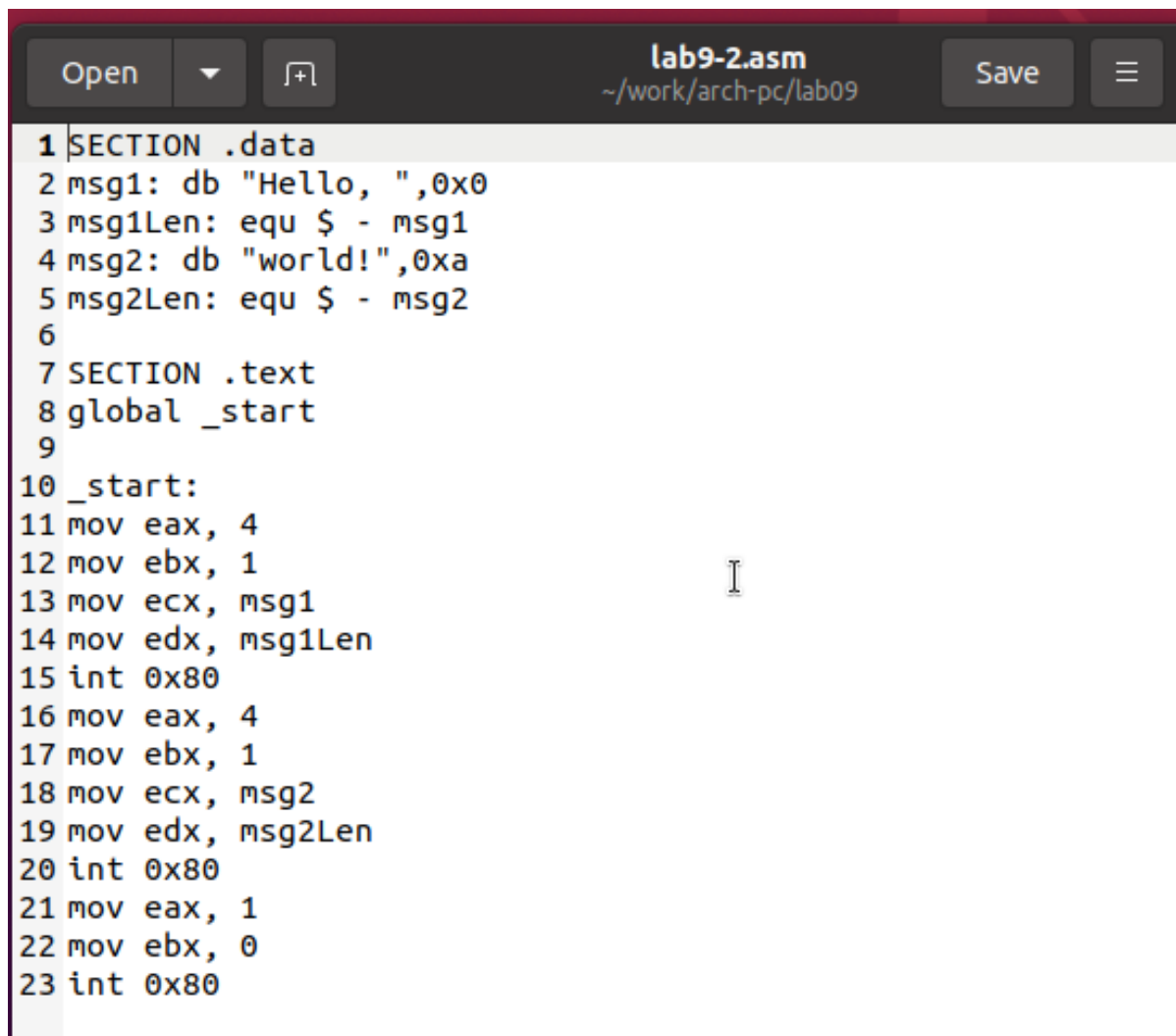
```

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

```

Рис. 2.4: Запуск программы lab9-1.asm

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



```

1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1Len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2Len: equ $ - msg2
6
7 SECTION .text
8 global _start
9
10 _start:
11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msg1Len
15 int 0x80
16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2Len
20 int 0x80
21 mov eax, 1
22 mov ebx, 0
23 int 0x80

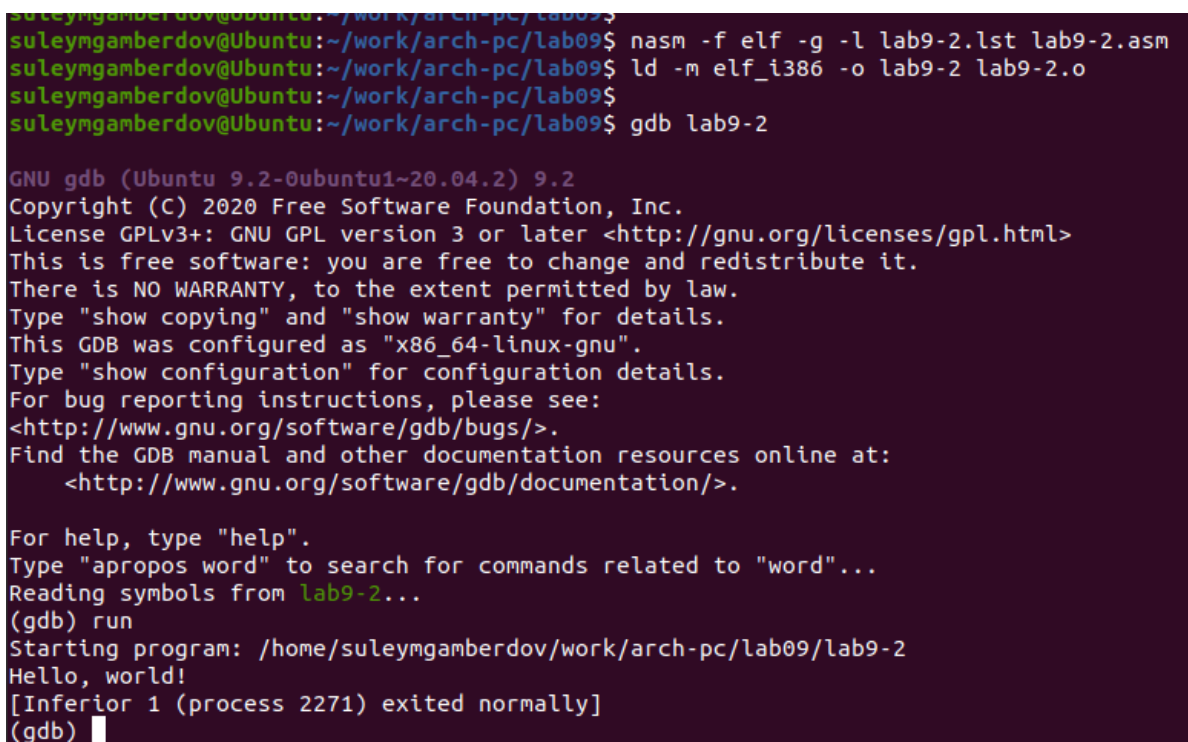
```

Рис. 2.5: Программа lab9-2.asm

Получил исполняемый файл. Для работы с GDB в исполняемый файл

необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'.

Загрузил исполняемый файл в отладчик gdb. Проверил работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r).



```
suleymgamberdov@ubuntu:~/work/arch-pc/lab09$  
suleymgamberdov@ubuntu:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm  
suleymgamberdov@ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o  
suleymgamberdov@ubuntu:~/work/arch-pc/lab09$ gdb lab9-2  
  
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2  
Copyright (C) 2020 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-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://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) run  
Starting program: /home/suleymgamberdov/work/arch-pc/lab09/lab9-2  
Hello, world!  
[Inferior 1 (process 2271) exited normally]  
(gdb) █
```

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

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

```
suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09
<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) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/suleymgamberdov/work/arch-pc/lab09/lab9-2

Breakpoint 1, 0x08049000 in _start ()
(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

```
suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09

0x08049014 <+20>:    int     $0x80
0x08049016 <+22>:    mov     $0x4,%eax
0x0804901b <+27>:    mov     $0x1,%ebx
0x08049020 <+32>:    mov     $0x804a008,%ecx
0x08049025 <+37>:    mov     $0x7,%edx
0x0804902a <+42>:    int     $0x80
0x0804902c <+44>:    mov     $0x1,%eax
0x08049031 <+49>:    mov     $0x0,%ebx
0x08049036 <+54>:    int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov     eax,0x4
    0x08049005 <+5>:    mov     ebx,0x1
    0x0804900a <+10>:   mov     ecx,0x804a000
    0x0804900f <+15>:   mov     edx,0x8
    0x08049014 <+20>:   int     0x80
    0x08049016 <+22>:   mov     eax,0x4
    0x0804901b <+27>:   mov     ebx,0x1
    0x08049020 <+32>:   mov     ecx,0x804a008
    0x08049025 <+37>:   mov     edx,0x7
    0x0804902a <+42>:   int     0x80
    0x0804902c <+44>:   mov     eax,0x1
    0x08049031 <+49>:   mov     ebx,0x0
    0x08049036 <+54>:   int     0x80
End of assembler dump.
(gdb) █
```

Рис. 2.8: Дизассемблированный код в режиме интел

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

```
suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd1b0 0xffffd1b0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags   0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       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 2278 In: _start L?? PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint     keep y   0x08049000  <_start>
       breakpoint     already hit 1 time
2      breakpoint     keep y   0x08049031  <_start+49>
(gdb) 
```

Рис. 2.9: Точка остановки

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

```
suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax    0x4      4      ecx    0x0      0
edx    0x0      0      ebx    0x0      0
esp    0xffffd1b0 0xffffd1b0  ebp    0x0      0x0
esi    0x0      0      edi    0x0      0
eip    0x8049005 0x8049005 <_start+5>  eflags  0x202    [ IF ]
cs     0x23     35     ss     0x2b     43
ds     0x2b     43     es     0x2b     43
fs     0x0      0      gs     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 2278 In: _start L?? PC: 0x8049005
edx    0x0      0
ebx    0x0      0
esp    0xffffd1b0 0xffffd1b0
ebp    0x0      0x0
esi    0x0      0
edi    0x0      0
eip    0x8049000 0x8049000 <_start>
eflags 0x202    [ IF ]
cs     0x23     35
ss     0x2b     43
ds     0x2b     43
es     0x2b     43
fs     0x0      0
gs     0x0      0
(gdb) si
0x08049005 in _start ()
(gdb) █
```

Рис. 2.10: Изменение регистров

```
suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd1b0  0xffffd1b0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016  0x8049016 <_start+22> eflags    0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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 2278 In: _start L?? PC: 0x8049016
cs      0x23      35
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
0x08049005 in _start ()
(gdb) si
0x0804900a in _start ()
(gdb) si
0x0804900f in _start ()
(gdb) si
0x08049014 in _start ()
(gdb) si
0x08049016 in _start ()
(gdb) 
```

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.

```

suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd1b0      0xffffd1b0      ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016      0x8049016 <_start+22>      eflags      0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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 2278 In: _start      L??      PC: 0x8049016
(gdb) si
0x0804900f in _start ()
(gdb) si
0x08049014 in _start ()
(gdb) si
0x08049016 in _start ()
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n"
(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"
(gdb)

```

Рис. 2.12: Изменение значения переменной

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



```
suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd1b0  0xffffd1b0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016  0x8049016 <_start+22> eflags   0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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 2278 In: _start L?? PC: 0x8049016
$7 = 0x8
$8 = 0x8
(gdb) p/s $eax
$9 = 8
(gdb) p/t $eax
$10 = 1000
(gdb) p/s $ecx
$11 = 134520832
(gdb) p/x $ecx
$12 = 0x804a000
(gdb) p/s $edx
$13 = 8
(gdb) p/t $edx
$14 = 1000
(gdb) p/x $edx
$15 = 0x8
(gdb) |
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx

```

suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8      ecx      0x804a000    134520832
edx      0x8      8      ebx      0x2            2
esp      0xffffd1b0 0xffffd1b0  ebp      0x0            0x0
esi      0x0      0      edi      0x0            0x0
eip      0x8049016 0x8049016 <_start+22>eflags 0x202          [ IF ]
cs       0x23     35      ss       0x2b           43
ds       0x2b     43      es       0x2b           43
fs       0x0      0      gs       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 2278 In: .start                                L??  PC: 0x8049016
(gdb) p/s $ecx
$11 = 134520832
(gdb) p/x $ecx
$12 = 0x804a000
(gdb) p/s $edx
$13 = 8
(gdb) p/t $edx
$14 = 1000
(gdb) p/x $edx
$15 = 0x8
(gdb) set $ebx=2
(gdb) p/s $ebx
$16 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$17 = 2
(gdb)

```

Рис. 2.14: Вывод значения регистра

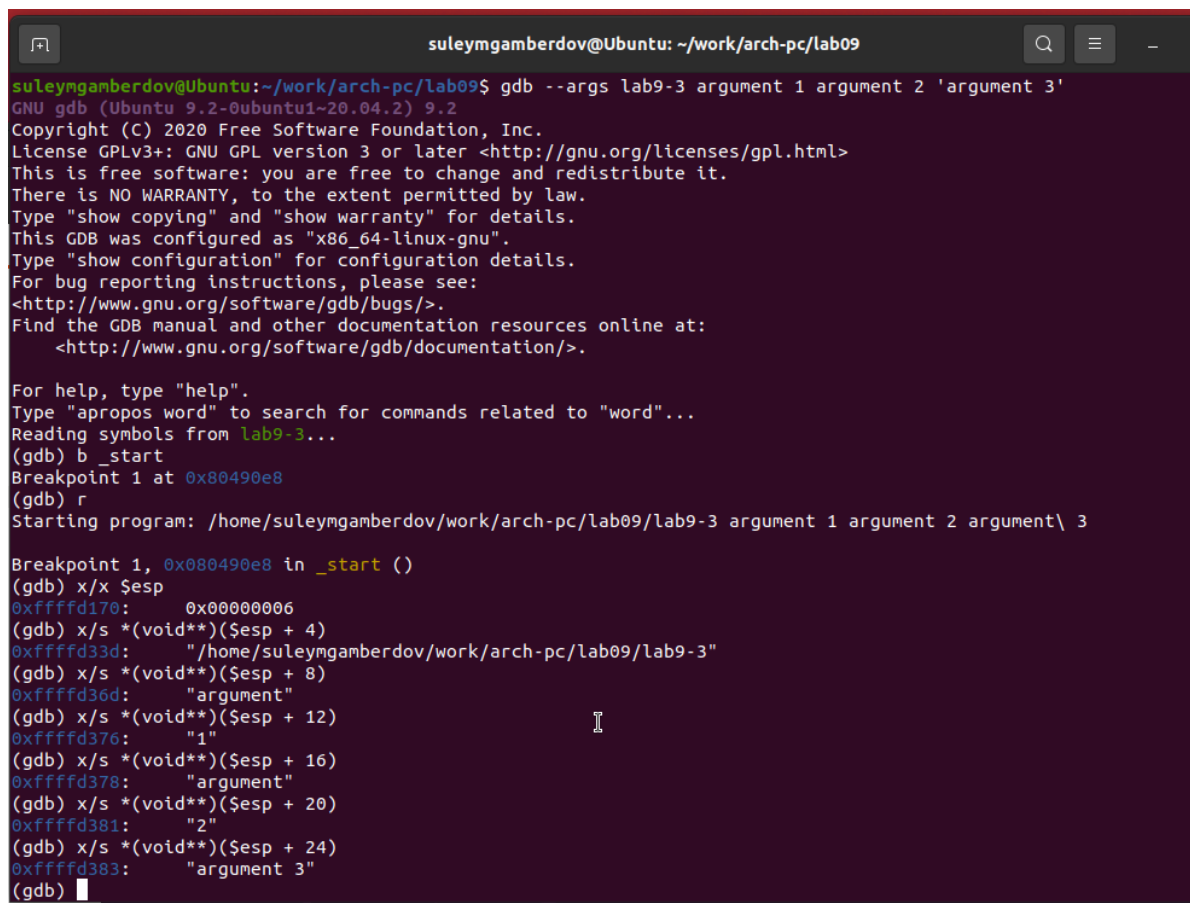
5. Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создал исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ `-args`. Загрузил исполняемый файл в отладчик, указав аргументы.

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.



```
suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09
suleymgamberdov@Ubuntu:~/work/arch-pc/lab09$ gdb --args lab9-3 argument 1 argument 2 'argument 3'
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2
Copyright (C) 2020 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

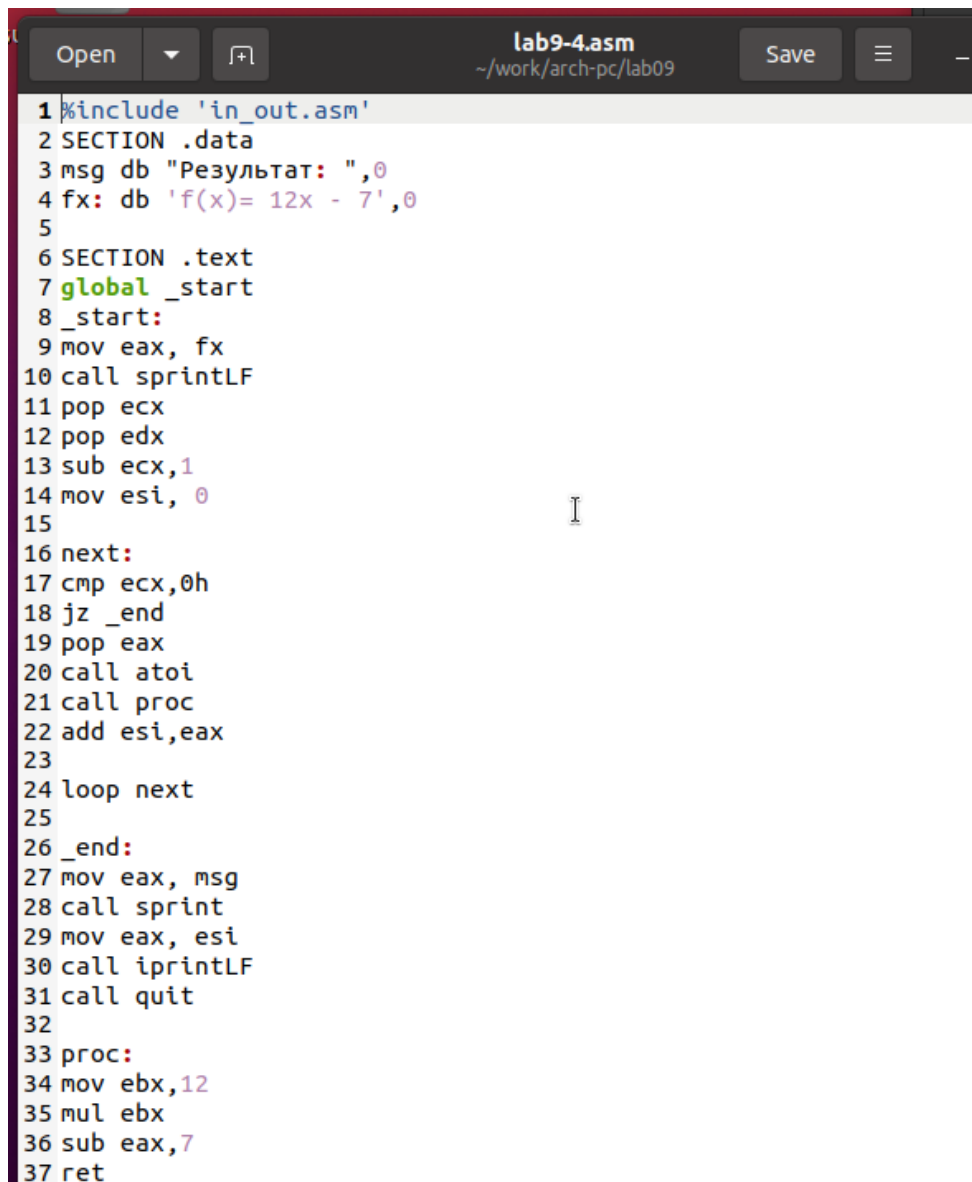
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /home/suleymgamberdov/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

Breakpoint 1, 0x80490e8 in _start ()
(gdb) x/x $esp
0xffffd170: 0x00000006
(gdb) x/s *(void**)($esp + 4)
0xffffd33d: "/home/suleymgamberdov/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd36d: "argument"
(gdb) x/s *(void**)($esp + 12)
0xffffd376: "1"
(gdb) x/s *(void**)($esp + 16)
0xffffd378: "argument"
(gdb) x/s *(void**)($esp + 20)
0xffffd381: "2"
(gdb) x/s *(void**)($esp + 24)
0xffffd383: "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

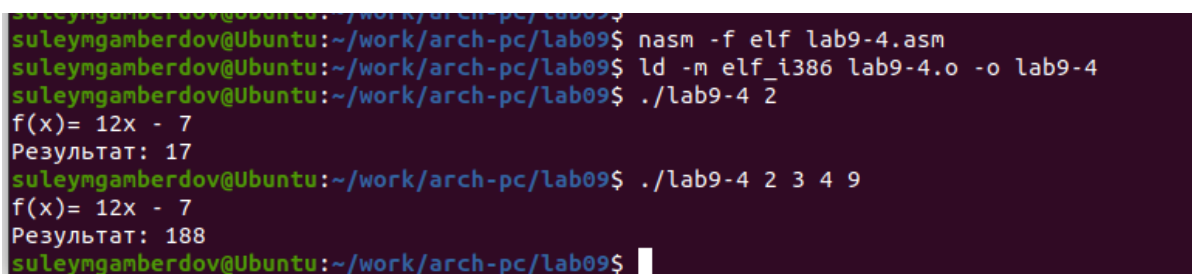
Объясню, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12]) - шаг равен размеру переменной - 4 байтам.

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



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 12x - 7',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 call proc
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprint
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 proc:
34 mov ebx,12
35 mul ebx
36 sub eax,7
37 ret
```

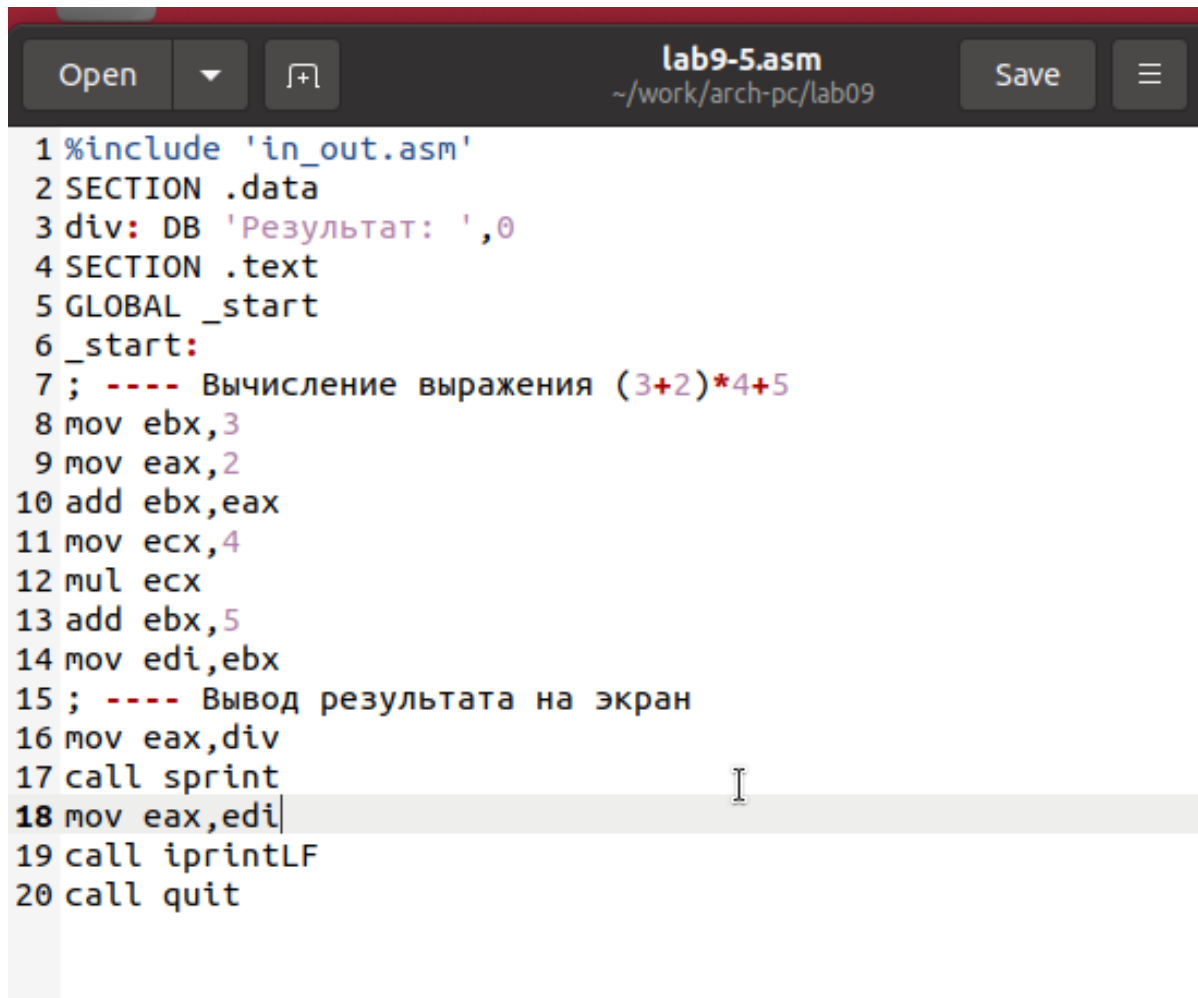
Рис. 2.16: Программа lab9-4.asm



```
suleymgamberdov@Ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
suleymgamberdov@Ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-4.o -o lab9-4
suleymgamberdov@Ubuntu:~/work/arch-pc/lab09$ ./lab9-4 2
f(x)= 12x - 7
Результат: 17
suleymgamberdov@Ubuntu:~/work/arch-pc/lab09$ ./lab9-4 2 3 4 9
f(x)= 12x - 7
Результат: 188
suleymgamberdov@Ubuntu:~/work/arch-pc/lab09$
```

Рис. 2.17: Запуск программы lab9-4.asm

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



```
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

```

suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09
eax      0x8      8      ecx      0x4      4
edx      0x0      0      ebx      0xa     10
esp      0xffffd1b0 0xffffd1b0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490fe 0x80490fe <_start+22>eflags 0x206 [ PF 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
B+ 0x80490e8 <_start+5> 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,0x5
>0x80490fb <_start+19> add ebx,0x5
0x80490fe <_start+22> mov edi,ebx04a000
0x8049100 <_start+24> mov eax,0x804a000rint>
0x8049105 <_start+29> call 0x804900f <sprint>
0x804910a <_start+34> mov eax,edi06 <iprintf>
0x804910c <_start+36> call 0x8049086 <iprintf>

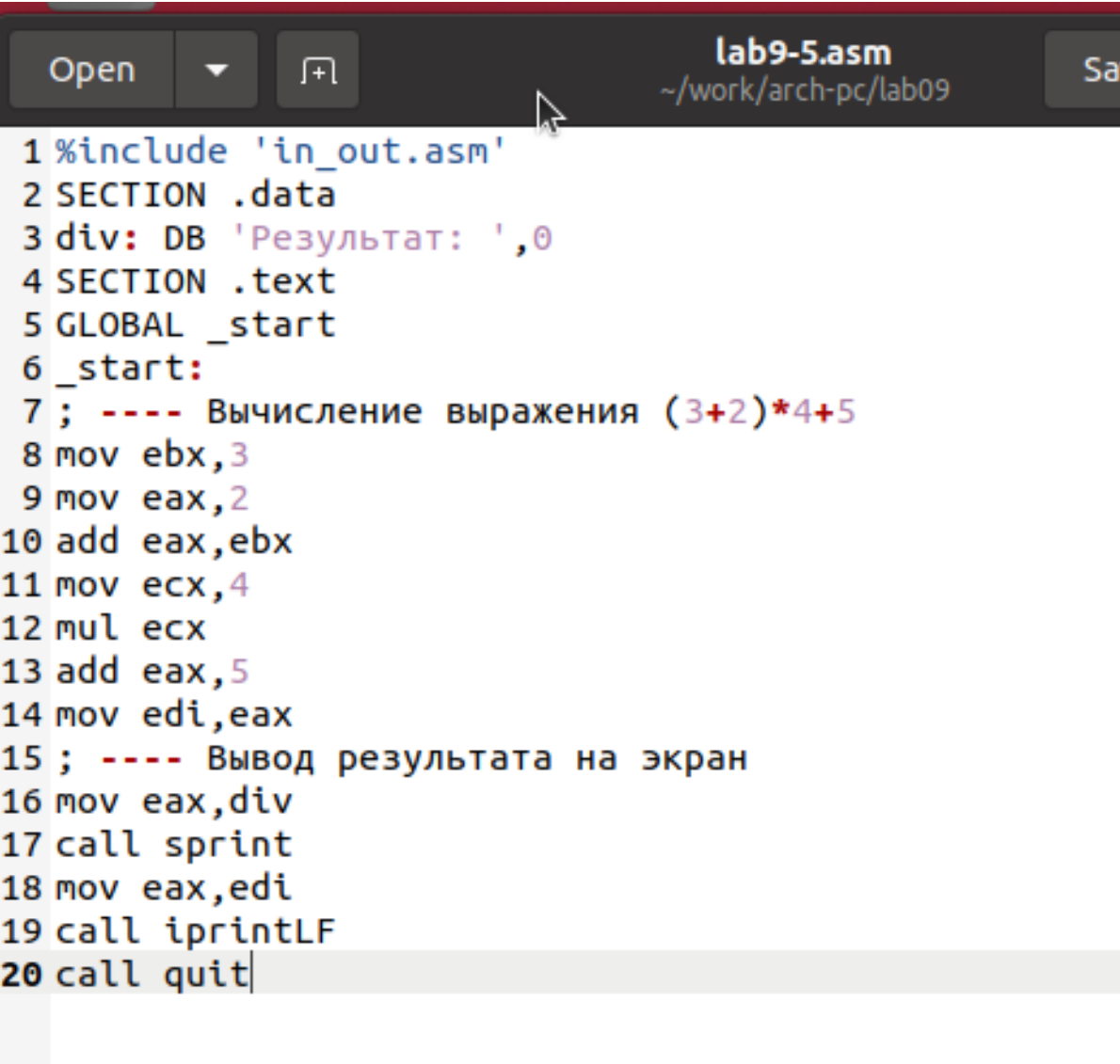
native process 2360 In: _start L?? PC: 0x80490fe
0x080490f4 No process In: L?? PC: ??
0x080490f4 in _start ()
(gdb) si
0x080490f9 in _start ()
(gdb) si
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 2360) exited normally]
(gdb)

```

Рис. 2.19: Отладка

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

Исправленный код программы



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

Рис. 2.20: Код исправлен

```

suleymgamberdov@Ubuntu: ~/work/arch-pc/lab09
eax    0x19      25      ecx    0x4        4
edx    0x0        0      ebx    0x3        3
esp    0xffffd1b0 0xffffd1b0  ebp    0x0        0x0
esi    0x0        0      edi    0x0        0
eip    0x80490fe 0x80490fe < _start+22>eflags 0x202      [ IF ]
cs     0x23      35      ss     0x2b      43
ds     0x2b      43      es     0x2b      43
fs     0x0        0      gs     0x0        0

0x80490e7 <_quit+12> ret
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
0x804910c <_start+36> call 0x8049086 <iprintf>

native process 2371 In: _start L?? PC: 0x80490fe
0x08049049 No process In: L?? PC: ??
0x080490f4 in _start ()
(gdb) si
0x080490f9 in _start ()
(gdb) si
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 2371) exited normally]
(gdb)

```

Рис. 2.21: Проверка работы



## **3 Выводы**

Освоили работу с подпрограммами и отладчиком.