

# **Лабораторная работа №9.**

**Понятие подпрограммы. Отладчик GDB.**

Митрофанов Тимур Александрович

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	23
4	Выводы	32

# Список иллюстраций

2.1	Создание подкаталога и файла . . . . .	6
2.2	Содержимое файла <i>lab09-1.asm</i> . . . . .	7
2.3	Создание исполняемого файла <i>lab09-1</i> и его запуск . . . . .	8
2.4	Изменённое содержимое файла <i>lab09-1.asm</i> . . . . .	9
2.5	Создание исполняемого файла и его запуск . . . . .	10
2.6	Создание файла <i>lab09-2.asm</i> . . . . .	10
2.7	Содержимое файла <i>lab09-2.asm</i> . . . . .	11
2.8	Создание исполняемого файла <i>lab09-2</i> . . . . .	11
2.9	Загрузка исполняемого файла в отладчик gdb и его запуск . . . . .	12
2.10	Установка брейкпойнта и запуск программы . . . . .	12
2.11	Ввод команды <code>***disassemble _start***</code> . . . . .	13
2.12	Включение на отображение команд с Intel'овским синтаксисом . . . . .	14
2.13	Режим псевдографики . . . . .	15
2.14	Режим псевдографики . . . . .	16
2.15	Работа с брейкпойнтами . . . . .	17
2.16	Выполнение инструкций . . . . .	18
2.17	Посмотр содержимого регистров . . . . .	19
2.18	Посмотр значения переменной <i>msg1</i> по имени . . . . .	19
2.19	Посмотр значения переменной <i>msg2</i> по адресу . . . . .	19
2.20	Изменение значения переменной <i>msg1</i> . . . . .	19
2.21	Изменение значения переменной <i>msg2</i> . . . . .	20
2.22	различные форматы значений регистра <i>edx</i> . . . . .	20
2.23	Изменение значения регистра <i>ebx</i> при помощи <code>set</code> . . . . .	20
2.24	Завершение работы с GDB . . . . .	21
2.25	Создание файла <i>lab09-3.asm</i> и преобразование его в исполняемый файл . . . . .	21
2.26	Загрузка файла <i>lab09-3.asm</i> в отладчик . . . . .	21
2.27	Установка брейкпойнта и запуск программы . . . . .	22
2.28	Проверка стека . . . . .	22
2.29	Проверка переменных в стеке . . . . .	22
3.1	Создание файла <i>lab09-4.asm</i> . . . . .	23
3.2	Содержимое файла <i>lab09-4.asm</i> . . . . .	24
3.3	Создание файла <i>lab09-4</i> и проверка его работы . . . . .	25
3.4	Создание файла <i>lab09-5.asm</i> . . . . .	26
3.5	Содержимое файла <i>lab09-5.asm</i> . . . . .	27
3.6	Создание файла <i>lab09-5</i> и его первоначальная отладка в GDB . . . . .	28

3.7	Пошаговое выполнение программы . . . . .	29
3.8	Содержимое файла <i>lab09-5.asm</i> . . . . .	30
3.9	Создание файла <i>lab09-5</i> и проверка его работы . . . . .	30

# 1 Цель работы


Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Выполнение лабораторной работы

Создал подкаталог *lab09*. В нём создал файл *touch lab09-1.asm* (рис. 2.1). В созданный файл ввёл текст листинга 9.1(рис. 2.2). Скомпелировал файл и проверил его работу (рис. 2.3).

```
tamtrofanov@tamtrofanov-VirtualBox:~$  
tamtrofanov@tamtrofanov-VirtualBox:~$ mkdir ~/work/arch-pc/lab09  
tamtrofanov@tamtrofanov-VirtualBox:~$ cd ~/work/arch-pc/lab09  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ touch lab09-1.asm  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.1: Создание подкаталога и файла

Открыть ▾ 

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

```
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 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ;-----
12 ; Основная программа
13 ;-----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax,x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax,result
23 call sprint
24 mov eax,[res]
25 call iprintLF
26 call quit
27 ;-----
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 _calcul:
31 mov ebx,2
32 mul ebx
33 add eax,7
34 mov [res],eax
35 ret ; выход из подпрограммы
```


Рис. 2.2: Содержимое файла *lab09-1.asm*

```
tamtrofanov@tamtrofanov-VirtualBox: ~/work/arch-pc/lab09$  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1  
Введите x: 1  
2x+7=9  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.3: Создание исполняемого файла *lab09-1* и его запуск

В соответствии с заданием создал подпрограмму которая отвечает за вычисления (рис. 2.4). Скомпелировал файл и проверил его работу (рис. 2.5).



Открыть ▾ 

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

```
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 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ;-----
12 ; Основная программа
13 ;-----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax, result
23 call sprint
24 mov eax, [res]
25 call iprintLF
26 call quit
27 ;-----
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 _calcul:
31 call _subcalcul
32 mov ebx, 2
33 mul ebx
34 add eax, 7
35 mov [res], eax
36 ret ; выход из подпрограммы
37 ;-----
38 _subcalcul:
39 mov ebx, 3
40 mul ebx,
41 dec eax
42 ret
```

Рис. 2.4: Изменённое содержимое файла *lab09-1.asm*

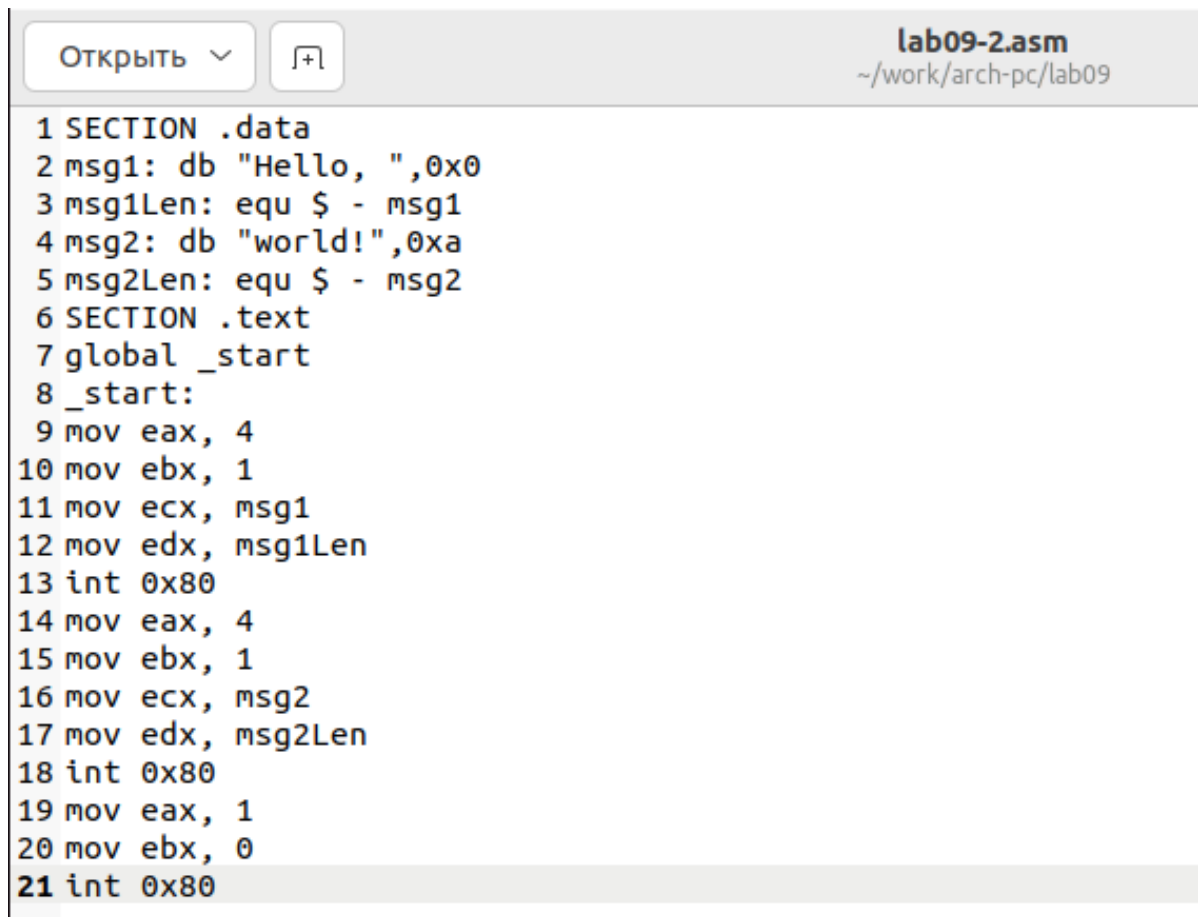
```
tamtrofanov@tamtrofanov-VirtualBox: ~/work/arch-pc/lab09$  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1  
Введите x: 1  
2x+7=11  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.5: Создание исполняемого файла и его запуск

Создал файл *lab09-2.asm* (рис. 2.6). В созданный файл ввёл текст листинга 9.2 (рис. 2.7). Для работы с GDB при создании исполняемого файла добавил ключ **-g** (рис. 2.8).

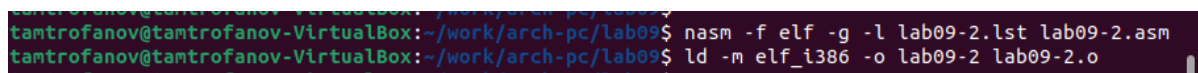
```
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ touch lab09-2.asm  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.6: Создание файла *lab09-2.asm*



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

Рис. 2.7: Содержимое файла *lab09-2.asm*



```
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
```

Рис. 2.8: Создание исполняемого файла *lab09-2*

Загрузил исполняемый файл в отладчик gdb и запустил его в нём (рис. 2.9).

```

tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-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 lab09-2...
(gdb) run
Starting program: /home/tamtrofanov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 32558) exited normally]
(gdb)

```

Рис. 2.9: Загрузка исполняемого файла в отладчик gdb и его запуск

Установил брейкпоинт на метку `_start` и запустил программу (рис. 2.10).

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/tamtrofanov/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)

```

Рис. 2.10: Установка брейкпоинта и запуск программы

Посмотрел дисассимилированный код программы с помощью команды *disassemble*, начиная с метки `***_start***` (рис. 2.11).

```
(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. 2.11: Ввод команды `***disassemble _start***`

Переключил на отображение команд с Intel'овским синтаксисом, введя команду *set disassembly-flavor intel* (рис. 2.12).

```

(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. 2.12: переключение на отображение команд с Intel'овским синтаксисом

***Перечисл различия отображения синтаксиса машинных команд в режимах ATТ и Intel.***

- В АТТ имена регистров начинаются с символа %, а имена операндов с \$, в то время как в intel используется привычный нам синтаксис.

Включил режим псевдографики для более удобного анализа программы (рис. 2.13).

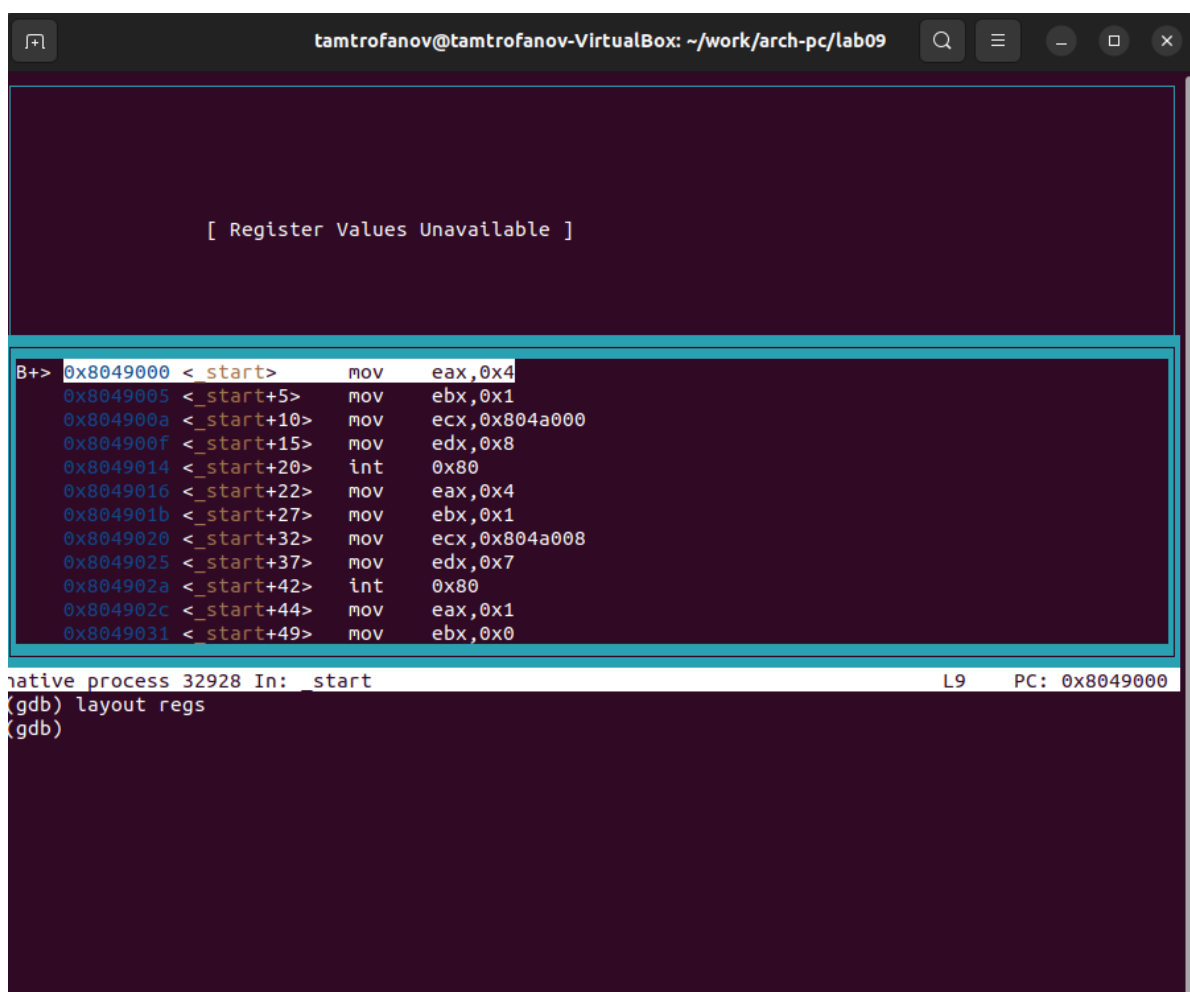


Рис. 2.13: Режим псевдографики

Получил данные о брейкпоинтах (рис. 2.14). Добавил ещё один брейкпоинт и проверил его наличие (рис. 2.15).

The screenshot shows a GDB debugger window with a dark background. At the top, a status bar displays the username 'tamtrofanov' and the file path '~/work/arch-pc/lab09'. Below this, a large panel shows the message '[ Register Values Unavailable ]'. A smaller panel below that displays assembly code with addresses and instructions. The code is as follows:

Address	Instruction
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
0x8049031	<_start+49> mov ebx,0x0

Below the assembly code, the GDB prompt shows the current state: 'native process 32928 In: \_start' with 'L9' and 'PC: 0x8049000'. The user has entered several commands: '(gdb) layout regs', '(gdb) info breakpoints', and '(gdb) info breakpoints'. The output shows that a breakpoint is set at address 0x08049000 in file lab09-2.asm at line 9, and it has been hit 1 time.

```
(gdb) layout regs
(gdb) info breakpoints
Undefined command: "info". Try "help".
Undefined command: "info". Try "help".
Undefined command: "info". Try "help".
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y   0x08049000  lab09-2.asm:9
breakpoint already hit 1 time
(gdb)
```

Рис. 2.14: Режим псевдографики



```
tamtrofanov@tamtrofanov-VirtualBox: ~/work/arch-pc/lab09 [ Register Values Unavailable ]

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
0x8049036 <_start+54> int 0x80
0x8049038 add BYTE PTR [eax],al

native process 32928 In: start L9 PC: 0x8049000
Undefined command: "infp". Try "help".
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab09-2.asm:20
(qdb)
```

Рис. 2.15: Работа с брейкпоинтами

Выполнил 5 инструкций с помощью команды stepi (рис. 2.16).

```

tamtrofanov@tamtrofanov-VirtualBox: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd220 0xffffd220
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
   0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0

native process 32928 In: _start L14 PC: 0x8049016
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint     keep y   0x08049000 lab09-2.asm:9
       breakpoint already hit 1 time
2      breakpoint     keep y   0x08049031 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 2.16: Выполнение инструкций

Посмотрел содержимое регистров при помощи команды `info registers` (рис. 2.17).

```
(gdb) i r
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd220 0xffffd220
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
(gdb) █
```

Рис. 2.17: Посмотр содержимого регистров

Посмотрел значение переменной msg1 по имени (рис. 2.18).

```
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) █
```

Рис. 2.18: Посмотр значения переменной msg1 по имени

Посмотрел значение переменной msg2 по адресу (рис. 2.19).

```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) █
```

Рис. 2.19: Посмотр значения переменной msg2 по адресу

Изменил первый символ переменной msg1 на **h** (рис. 2.20).

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) █
```

Рис. 2.20: Изменение значения переменной msg1

Изменил первый символ переменной msg2 на **Q** (рис. 2.21).

```
(gdb) set {char}&msg2='Q'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "Qor!d!\n\034"
```

Рис. 2.21: Изменение значения переменной msg2

Выведел в различных форматах значение регистра edx (рис. 2.22).

```
(gdb) p/x $edx
$1 = 0x8
(gdb) p/t $edx
$2 = 1000
(gdb) p/c $edx
$3 = 8 '\b'
(gdb)
```

Рис. 2.22: различные форматы значений регистра edx

С помощью команды **set** измените значение регистра ebx (рис. 2.23).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb)
```

Рис. 2.23: Изменение значения регистра ebx при помощи set

**Объясните разницу вывода команд p/s \$ebx.**

- В первом случае мы переводим символ в строковой вид.

Завершил выполнение программы с помощью команды continue и вышел из GDB с помощью команды quit (рис. 2.24).

```

(gdb) continue
Continuing.
Qorld!
Breakpoint 2, _start () at lab09-2.asm:20
(gdb) continue
Continuing.
[Inferior 1 (process 32928) exited normally]
(gdb) quit

```

Рис. 2.24: Завершение работы с GDB

Скопировал файл *lab8-2.asm* в файл с именем *lab09-3.asm* и создал исполняемый файл (рис. 2.25).

```

tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o

```

Рис. 2.25: Создание файла *lab09-3.asm* и преобразование его в исполняемый файл

Загрузил исполняемый файл в отладчик, указав аргументы (рис. 2.26).

```

tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-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 lab09-3

```

Рис. 2.26: Загрузка файла *lab09-3.asm* в отладчик

установил точку остановки перед первой инструкцией в программе и запустил ее (рис. 2.27).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/tanttrofanov/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3
Breakpoint 1, _start () at lab09-3.asm:5
5      pop есх ; Извлекаем из стека в `есх` количество
(gdb)

```

Рис. 2.27: Установка брейкпойнта и запуск программы

Проверил сколько в стеке хранится переменных (рис. 2.28).

```

(gdb) x/x $esp
0xffffd1e0:      0x00000005
(gdb)

```

Рис. 2.28: Проверка стека

Посмотрел остальные позиции стека (рис. 2.29).

```

(gdb) x/s *(void**)($esp + 4)
0xffffd393:      "/home/tanttrofanov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd3c0:      "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xffffd3d2:      "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xffffd3e3:      "2"
(gdb) x/s *(void**)($esp + 20)
0xffffd3e5:      "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
(gdb)

```

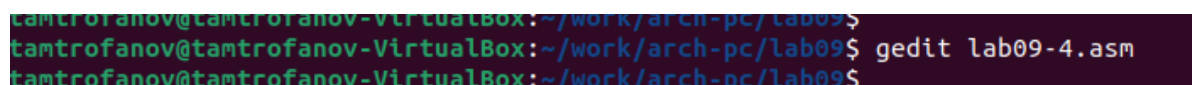
Рис. 2.29: Проверка переменных в стеке

**Объясните, почему шаг изменения адреса равен 4 ( $[esp+4]$ ,  $[esp+8]$ ,  $[esp+12]$  и т.д.)**

- Так как значение количества аргументов равно 4


### 3 Задание для самостоятельной работы

Создал файл *lab09-4.asm* (рис. 3.1). Перенёс программу из лабораторной работы № 8 в файл и изменил в соответствии с заданием (рис. 3.2). Создал исполнительный файл и проверил его работу (рис. 3.3).



```
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ gedit lab09-4.asm  
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 3.1: Созлание файла *lab09-4.asm*

Открыть ▾ 

lab09-4.asm  
~/work/arch-pc/lab09

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5
6 SECTION .text
7
8 global _start
9 _start:
10 pop ecx
11 pop edx
12 sub ecx,1
13 mov esi,0
14
15 next:
16 cmp ecx,0h
17 jz _end
18
19 pop eax
20 call atoi
21 call proga
22 loop next
23
24 _end:
25 mov eax, msg
26 call sprint
27 mov eax, esi
28 call iprintLF
29 call quit
30
31 proga:
32 mov ebx,10
33 mul ebx
34 sub eax,5
35 add esi,eax
36 ret
```

Рис. 3.2: Содержимое файла *lab09-4.asm*



```

tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-4 1 2 3 4
Результат: 80
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$

```

Рис. 3.3: Создание файла *lab09-4* и проверка его работы

### ***КОД ПРОГРАММЫ***

```

#include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

SECTION .text

global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0

next:
cmp ecx,0h
jz _end

pop eax
call atoi
call proga
loop next

```

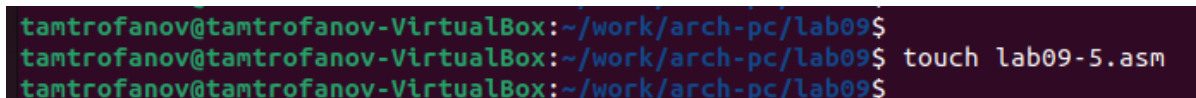
```

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

proga:
mov ebx, 10
mul ebx
sub eax, 5
add esi, eax
ret

```

Создал файл *lab09-5.asm* (рис. 3.4). Внёс код из листинга 9.3 в файл (рис. 3.5). Создал исполняемый файл с применением отладчика, загрузил в него этот файл, подготовил программу для работы в отладчике (поставил брейкпойт, подключил граф. интр.) (рис. 3.6).

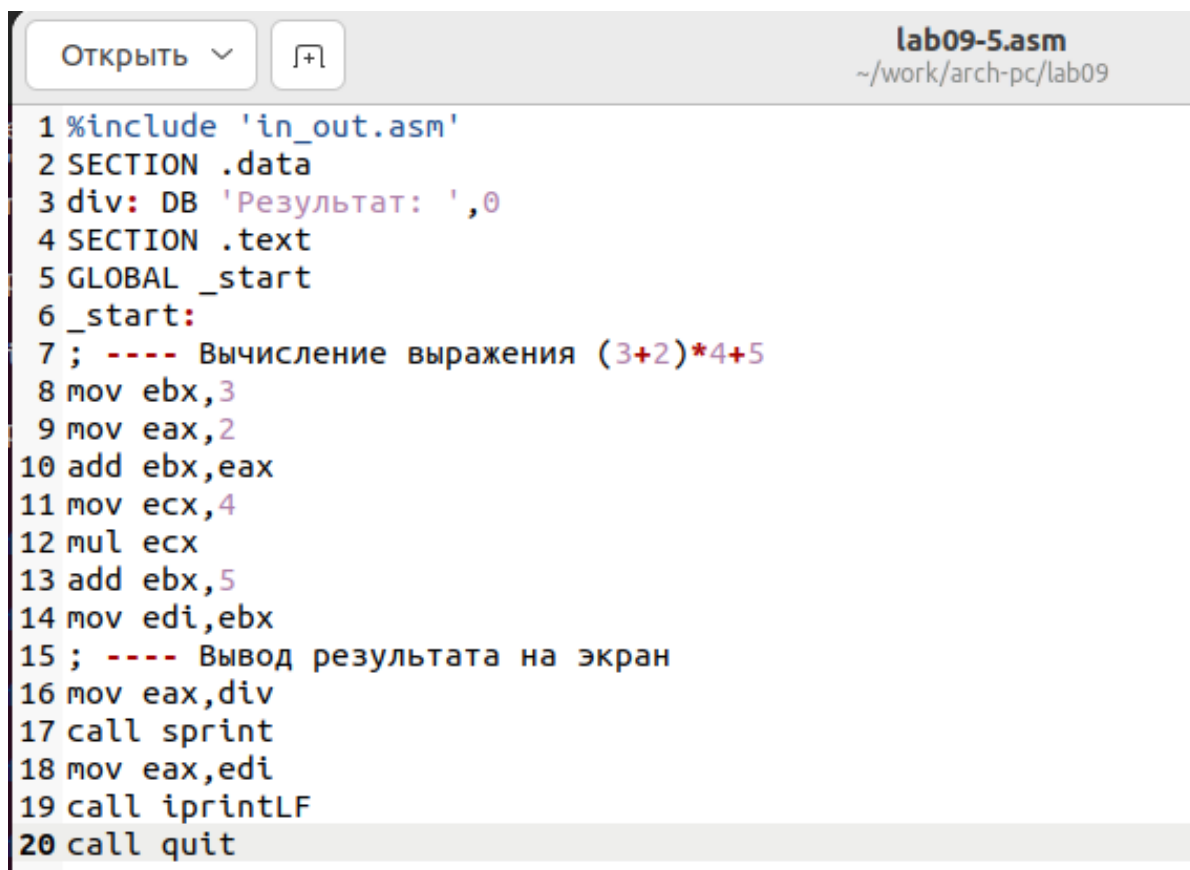


```

tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ touch lab09-5.asm
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$

```

Рис. 3.4: Создание файла *lab09-5.asm*



```
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. 3.5: Содержимое файла *lab09-5.asm*

```

tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-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 lab09-5...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-5.asm, line 8.
(gdb) run
Starting program: /home/tamtrofanov/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:8
8      mov ebx,3
(gdb) layout asm

```

Рис. 3.6: Создание файла *lab09-5* и его первоначальная отладка в GDB

Пошагово выполняя программу нашел ошибку в использовании неправильного регистра при умножении (рис. 3.7).

```
tamtrofanov@tamtrofanov-VirtualBox: ~/work/arch-pc/lab09

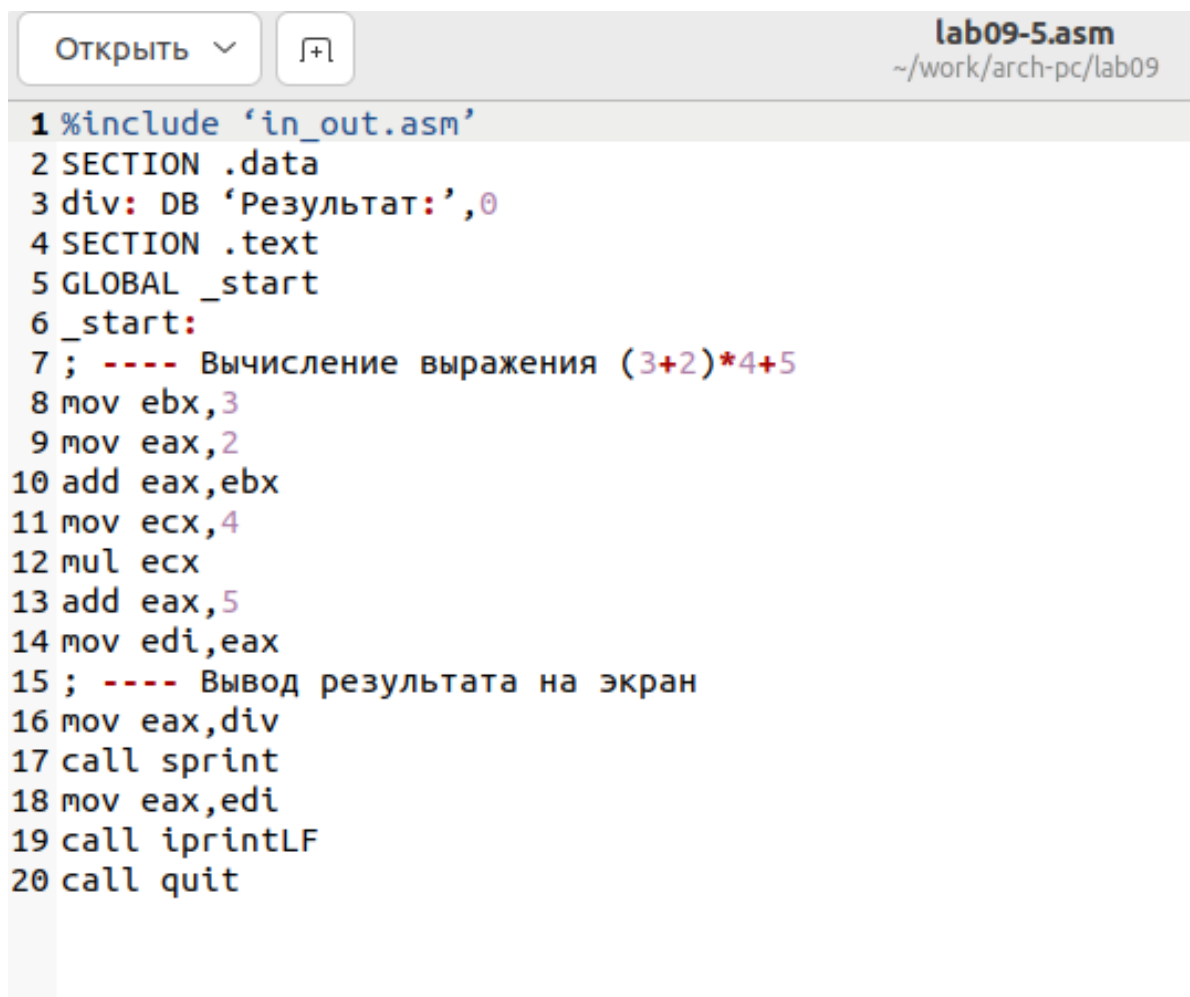
Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffd220 0xffffd220
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490fb 0x80490fb < start+19>
eflags   0x206    [ PF IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0

B+ 0x80490e8 <_start>    mov    $0x3,%ebx
0x80490ed <_start+5>    mov    $0x2,%eax
0x80490f2 <_start+10>   add    %eax,%ebx
0x80490f4 <_start+12>   mov    $0x4,%ecx
0x80490f9 <_start+17>   mul    %ecx
> 0x80490fb <_start+19> add    $0x5,%ebx
0x80490fe <_start+22>   mov    %ebx,%edi
0x8049100 <_start+24>   mov    $0x804a000,%eax
0x8049105 <_start+29>   call   0x804900f <sprint>
0x804910a <_start+34>   mov    %edi,%eax
0x804910c <_start+36>   call   0x8049086 <iprintLF>
0x8049111 <_start+41>   call   0x80490db <quit>
0x8049116      add    %al,(%eax)
0x8049118      add    %al,(%eax)
0x804911a      add    %al,(%eax)
0x804911c      add    %al,(%eax)
0x804911e      add    %al,(%eax)

native process 35028 In: _start L13 PC: 0x80490fb
(gdb) layout regs
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 3.7: Пошаговое выполнение программы

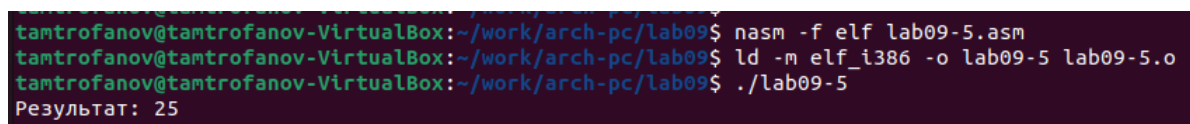
Внёс необходимые изменения в файл *lab09-5.asm* (рис. 3.8). Создал исполнимый файл и проверил его работу (рис. 3.9).



```
lab09-5.asm
~/work/arch-pc/lab09

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. 3.8: Содержимое файла *lab09-5.asm*



```
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
tamtrofanov@tamtrofanov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
```

Рис. 3.9: Создание файла *lab09-5* и проверка его работы

### **КОД ПРОГРАММЫ**

```
%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

```

## 4 Выводы

Сегодня я приобрёл навыки написания программ с использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями.