

# **Отчёта по лабораторной работе №9**

**Понятие подпрограммы. Отладчик GDB.**

Попутников Егор Сергеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
2.1	Реализация подпрограмм в NASM . . . . .	5
2.2	Отладка программ с помощью GDB . . . . .	8
<b>3</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

2.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code> . . . . .	5
2.2	Заполняем файл . . . . .	6
2.3	Запускаем файл и проверяем его работу . . . . .	6
2.4	Изменяем файл, добавляя еще одну подпрограмму . . . . .	7
2.5	Запускаем файл и смотрим на его работу . . . . .	7
2.6	Создаем файл . . . . .	8
2.7	Загружаем исходный файл в отладчик . . . . .	9
2.8	Запускаем программу командой <code>run</code> . . . . .	9
2.9	Запускаем программу с брейкпоинтом . . . . .	9
2.10	Смотрим дисассимилированный код программы . . . . .	10
2.11	Переключаемся на синтаксис Intel . . . . .	10
2.12	Включаем отображение регистров, их значений и результат дисассимилирования программы . . . . .	11
2.13	Используем команду <code>info breakpoints</code> и создаем новую точку останова . . . . .	11
2.14	Смотрим информацию . . . . .	12
2.15	Отслеживаем регистры . . . . .	12
2.16	Смотрим значение переменной . . . . .	12
2.17	Смотрим значение переменной . . . . .	13
2.18	Меняем символ . . . . .	13
2.19	Меняем символ . . . . .	13
2.20	Смотрим значение регистра . . . . .	13
2.21	Прописываем команды <code>c</code> и <code>quit</code> . . . . .	14
2.22	Копируем файл . . . . .	14
2.23	Создаем и запускаем в отладчике файл . . . . .	14
2.24	Устанавливаем точку останова . . . . .	15
2.25	Изучаем полученные данные . . . . .	15
2.26	Проверяем работу программы . . . . .	16
2.27	Создаем и смотрим на работу программы(работает неправильно) . . . . .	16
2.28	Ищем ошибку регистров в отладчике . . . . .	16
2.29	Меняем файл . . . . .	17
2.30	Создаем и запускаем файл(работает корректно) . . . . .	17

# 1 Цель работы

Познакомиться с методами отладки при помощи GDB, его возможностями.

## 2 Выполнение лабораторной работы

### 2.1 Реализация подпрограмм в NASM

Создаем каталог для программ лабораторной работы, и в нем создаем файл (рис. 2.1).

```
egor@espoputnikov-dk3n56:~$ mkdir ~/work/arch-pc/lab09
egor@espoputnikov-dk3n56:~$ cd ~/work/arch-pc/lab09
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ touch lab09-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$
```

Рис. 2.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.1 (рис. 2.2).

```

#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления

```

Рис. 2.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. 2.3).

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ gedit lab09-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 2
2x+7=11

```

Рис. 2.3: Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив подпрограмму в подпрограмму(по условию) (рис. 2.4).

```

1 %include 'in_out.asm'
2 SECTION .data
3     msg: DB 'Введите x: ',0
4     result: DB '2(3x-1)+7=',0
5 SECTION .bss
6     x: RESB 80
7     res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11     mov eax, msg
12     call sprint
13     mov ecx, x
14     mov edx, 80
15     call sread
16     mov eax,x
17     call atoi
18     call _calcul
19     mov eax,result
20     call sprint
21     mov eax,[res]
22     call iprintLF
23     call quit
24     _calcul:
25         call _subcalcul
26         mov ebx,2
27         mul ebx
28         add eax,7

```

Рис. 2.4: Изменяем файл, добавляя еще одну подпрограмму

Создаем исполняемый файл и запускаем его (рис. 2.5).

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ gedit lab09-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 2
2(3x-1)+7=17
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$

```

Рис. 2.5: Запускаем файл и смотрим на его работу

## 2.2 Отладка программ с помощью GDB

Создаем новый файл в каталоге Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.2 (рис. 2.6).

```
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. 2.6: Создаем файл

Получаем исходный файл с использованием отладчика gdb (рис. 2.7).



```

egor@espoputnikov-dk3n56:~/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)

```

Рис. 2.7: Загружаем исходный файл в отладчик

Запускаем команду в отладчике (рис. 2.8).

```

(gdb) run
Starting program: /home/egor/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 5006) exited normally]
(gdb)

```

Рис. 2.8: Запускаем программу командой run

Устанавливаем брейкпоинт на метку `_start` и запускаем программу (рис. 2.9).

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb)

```

Рис. 2.9: Запускаем программу с брейкпоином

Смотрим дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start` (рис. 2.10).

```
(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. 2.10: Смотрим дисассимилированный код программы

Переключаемся на отображение команд с Intel'овским синтаксисом (рис. 2.11).

```
(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. 2.11: Переключаемся на синтаксис Intel

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel:

- 1.Порядок операндов: В АТТ синтаксисе порядок операндов обратный, сначала указывается исходный операнд, а затем - результирующий операнд. В Intel синтаксисе порядок обычно прямой, результирующий операнд указывается первым, а исходный - вторым.
- 2.Разделители: В АТТ синтаксисе разделители операндов - запятые. В Intel синтаксисе разделители могут быть запятые или косые черты (/).
- 3.Префиксы размера операндов: В АТТ синтаксисе размер операнда указывается перед операндом с использованием префиксов, таких как "b" (byte), "w" (word),

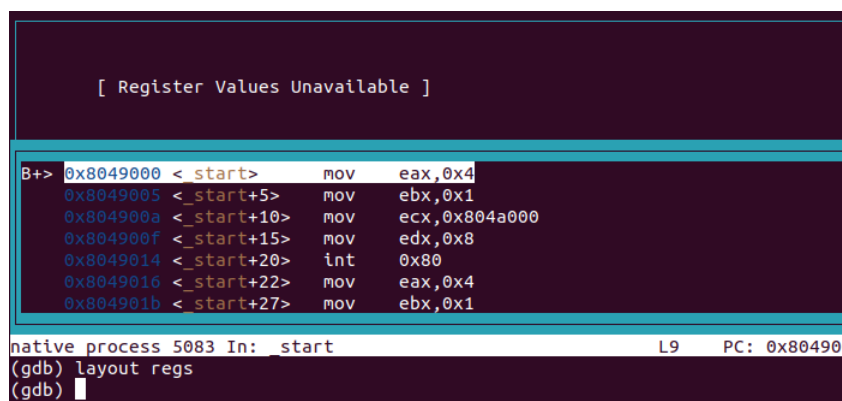
“l” (long) и “q” (quadword). В Intel синтаксисе размер операнда указывается после операнда с использованием суффиксов, таких как “b”, “w”, “d” и “q”.

4.Знак операндов: В АТТ синтаксисе операнды с позитивными значениями предваряются символом “*.Intel*”.

5.Обозначение адресов: В АТТ синтаксисе адреса указываются в круглых скобках. В Intel синтаксисе адреса указываются без скобок.

6.Обозначение регистров: В АТТ синтаксисе обозначение регистра начинается с символа “%”. В Intel синтаксисе обозначение регистра может начинаться с символа “R” или “E” (например, “%eax” или “RAX”).

Включаем режим псевдографики (рис. 2.12).



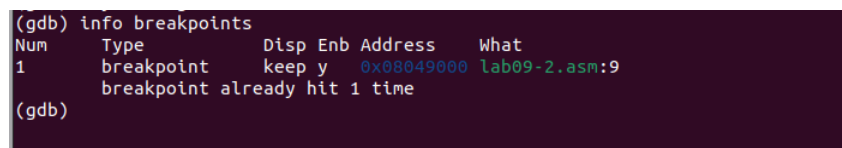
The screenshot shows a debugger window with a dark background. At the top, a status bar reads "[ Register Values Unavailable ]". Below it, a list of assembly instructions is displayed with their addresses and disassembled forms:

Address	Disassembly
0x8049000 <_start>	mov eax,0x4
0x8049005 <_start+5>	mov ebx,0x1
0x804900a <_start+10>	mov ecx,0x804a000
0x804900f <_start+15>	mov edx,0x8
0x8049014 <_start+20>	int 0x80
0x8049016 <_start+22>	mov eax,0x4
0x804901b <_start+27>	mov ebx,0x1

Below the assembly list, the status bar shows "native process 5083 In: \_start" and "L9 PC: 0x80490". At the bottom, the command prompt shows "(gdb) layout regs" and "(gdb) |".

Рис. 2.12: Включаем отображение регистров, их значений и результат дисассимилирования программы

Проверяем была ли установлена точка останова и устанавливаем точку останова предпоследней инструкции (рис. 2.13).



The screenshot shows a debugger window with a dark background. The command prompt shows "(gdb) info breakpoints". Below it, the output of the command is displayed:

Num	Type	Disp	Enb	Address	What
1	breakpoint	keep y		0x08049000	lab09-2.asm:9

Below the table, the output continues with "breakpoint already hit 1 time". At the bottom, the command prompt shows "(gdb)".

Рис. 2.13: Используем команду info breakpoints и создаем новую точку останова

Посмотрим информацию о всех установленных точках останова (рис. 2.14).

```
(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)
```

Рис. 2.14: Смотрим информацию

Выполняем 5 инструкций командой si (рис. 2.15).

```
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0c0 0xffffd0c0
ebp      0x0      0
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

native process 6117 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. 2.15: Отслеживаем регистры

Во время выполнения команд менялись регистры: ebx, ecx, edx, eax, eip. Смотрим значение переменной msg1 по имени (рис. 2.16).

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

Рис. 2.16: Смотрим значение переменной

Смотрим значение переменной msg2 по адресу (рис. 2.17).

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

Рис. 2.17: Смотрим значение переменной

Изменим символы переменной msg1 (рис. 2.18).

```
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hhlllo, "
(gdb)
```

Рис. 2.18: Меняем символ

Изменим символы переменной msg2 (рис. 2.19).

```
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>: "Lor d!\n\034"
(gdb)
```

Рис. 2.19: Меняем символ

Смотрим значение регистра edx в разных форматах (рис. 2.20).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$2 = 2
(gdb)
```

Рис. 2.20: Смотрим значение регистра

Выводится разные значения, так как команда без кеавычек присваивает регистру вводимое значение.

Прописываем команды для завершения программы и выхода из GDB (рис. 2.21).

```
(gdb) continue
Continuing.
hlllo, Lor d!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb) stepi
(gdb) quit
```

Рис. 2.21: Прописываем команды с и quit

Копируем файл lab8-2.asm в файл с именем lab09-3.asm (рис. 2.22).

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm
~/work/arch-pc/lab09/lab09-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab
09-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3
.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$
```

Рис. 2.22: Копируем файл

Создаем исполняемый файл и запускаем его в отладчике GDB (рис. 2.23).

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргу
мент 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...
(gdb)
```

Рис. 2.23: Создаем и запускаем в отладчике файл

Установим точку останова перед первой инструкцией в программе и запустим ее (рис. 2.24).

```

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

```

Рис. 2.24: Устанавливаем точку останова

Смотрим позиции стека по разным адресам (рис. 2.25).

```

(gdb) x/x $esp
0xffffd220: 0x00000005
(gdb)
(gdb) x/s *(void**)(esp + 4)
0xffffd3d4: "/home/egor/work/arch-pc/lab09/lab09-3"
(gdb)
(gdb) x/s *(void**)(esp + 4)
0xffffd3d4: "/home/egor/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3fa: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd40c: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd41d: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd41f: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>

```

Рис. 2.25: Изучаем полученные данные

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

##Задание для самостоятельной работы

###Задание 1

Копируем файл lab8-4.asm(ср №1 в ЛБ8) Открываем файл в Midnight Commander и меняем его, создавая подпрограмму Создаем исполняемый файл и запускаем его (рис. 2.26).

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4
.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ./lab09-4
Введите x: 3
2(x-1)=4
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$

```

Рис. 2.26: Проверяем работу программы

### ###Задание 2

Создаем новый файл в директории Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.3 Создаем исполняемый файл и запускаем его (рис. 2.27).

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5
.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$

```

Рис. 2.27: Создаем и смотрим на работу программы(работает неправильно)

Создаем исполняемый файл и запускаем его в отладчике GDB и смотрим на изменение регистров командой si (рис. 2.28).

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab
09-5.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5
.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ gdb --args lab09-3
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...
(gdb)

```

Рис. 2.28: Ищем ошибку регистров в отладчике

Изменяем программу для корректной работы (рис. 2.29).



```

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 eax,3
9 mov ebx,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 iprintlnLF
20 call quit

```

Рис. 2.29: Меняем файл

Создаем исполняемый файл и запускаем его (рис. 2.30).

```

egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
egor@espoputnikov-dk3n56:~/work/arch-pc/lab09$

```

Рис. 2.30: Создаем и запускаем файл(работает корректно)

## 3 Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.