

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Семенченко Т. С.

Группа: НКАбд-05-25

МОСКВА

2025 г.

Оглавление

1 Цель работы.....	3
2 Задачи.....	4
3 Выполнение лабораторной работы	5
3.1 Реализация подпрограмм в NASM.....	5
3.2 Отладка программ с помощью GDB.....	5
3.3 Добавление точек останова	8
3.4 Работа с данными программы в GDB	10
3.5 Обработка аргументов командной строки в GDB	14
4 Задания для самостоятельной работы	16
5 Вывод	20

1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм.
Познакомиться с методами отладки при помощи GDB и его основными возможностями.

2 Задачи

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация подпрограмм в NASM

Создала каталог для выполнения лабораторной работы №9, перешла в него и создала файл lab9-1.asm. Копирую в файл код из первого листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. 3.1.1)

```
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 7
2x+7=21
```

Рис. 3.1.1 Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. 3.1.2)

```
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 7
2(3x-1)+7=47
```

Рис. 3.1.2 Запуск исправленной программы

3.2 Отладка программ с помощью GDB

Создала файл lab9-2.asm, записала в нее текст программы из второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике (рис. 3.2.1)

```

tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
tssemenchenko@dk6n18 ~/work/arch-pc/lab09 $ gdb lab9-2
Exception caught while booting Guile.
Error in function "load-thunk-from-memory":
missing DT_GUILE_ENTRY
gdb: warning: Could not complete Guile gdb module initialization from:
/usr/share/gdb/guile/gdb/boot.scm.
Limited Guile support is available.
Suggest passing --data-directory=/path/to/gdb/data-directory.
GNU gdb (Gentoo 16.3 vanilla) 16.3
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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...

```

Рис. 3.2.1 Запуск программы в отладчике

Запустив программу командой `run`, я убедилась в том, что она работает исправно (рис. 3.2.2)

```

(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/t/s/tssemenchenko/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 3490) exited normally]

```

Рис. 3.2.2 Проверка программы

Для более подробного анализа программы установила брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустила её (рис. 3.2.3)

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.

```

Рис. 3.2.3 Запуск программы отладчика с установленным брейкпоинтом

Посмотрела дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 3.2.4)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/t/s/tssemenchenko/work/arch-pc/lab09/
b9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      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.
```

Рис. 3.2.4 Дисассимидированный код программы


Переключилась на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 3.2.5)

```
(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.
```

Рис. 3.2.5 Демонстрация запуска программы

Различия между синтаксисом АТТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как ах, еах, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

Включаю режим псевдографики для более удобного анализа программы (рис. 3.2.6)



```
tssemenchenko@dk6n18 - lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      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

native process 3499 (asm) In: _start      L9      PC: 0x8049000
(gdb) layout regs
(gdb) □
```

Рис. 3.2.6 Режим псевдографики

3.3 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 3.3.1)

```
tssemenchenko@dk7n08 - lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc330 0xffffc330
ebp      0x0      0x0
esi      0x0      0

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
0x804903a add BYTE PTR [eax],al

native process 3045 (asm) In: _start L9 PC:
(gdb) layout regs
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint      keep y  0x08049000 lab9-2.asm:9
        breakpoint already hit 1 time
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) □
```

Рис. 3.3.1 Список брекпоинтов

Устанавливаю еще одну точку останова по адресу инструкции (рис. 3.3.2)

```
tssemenchenko@dk7n08 - lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc330 0xffffc330
ebp      0x0      0x0
esi      0x0      0

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
0x804903a      add BYTE PTR [eax],al

native process 3045 (asm) In: _start L9 PC:
breakpoint already hit 1 time
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y 0x08049000 lab9-2.asm:9
breakpoint already hit 1 time
2        breakpoint keep y 0x08049031 lab9-2.asm:20
(gdb) □
```

Рис. 3.3.2 Добавление второй точки останова

3.4 Работа с данными программы в GDB

Просматриваю содержимое регистров командой `info registers`. Смотрю содержимое переменных по имени и по адресу (рис. 3.4.1)

```
tssemenchenko@dk7n08 - lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc330 0xffffc330
ebp      0x0      0x0
esi      0x0      0

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
0x804903a add BYTE PTR [eax],al

native process 3045 (asm) In: _start L9 PC:
esi      0x0      0
edi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) □
```

Рис. 3.4.1 Просмотр содержимого регистров

Меняю содержимое переменных по имени и по адресу (рис. 3.4.2)

+

tssemenchenko@dk7n08 - lab09

Q

≡

Register group: general

eax	0x0	0
ecx	0x0	0
edx	0x0	0
ebx	0x0	0
esp	0xffffc330	0xffffc330
ebp	0x0	0x0
esi	0x0	0

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

0x804903a

add

BYTE PTR [eax],al

native process 3045 (asm) In: _start

L9

PC:

(gdb) set {char}msg1='h'

'msg1' has unknown type; cast it to its declared type

(gdb) set {char}&msg1='h'

(gdb) x/1sb &msg1

0x804a000 <msg1>: "hello, "

(gdb) set {char}&msg2='x'

(gdb) x/1sb &msg2

0x804a008 <msg2>: "xor!d!\n\034"

(gdb)

Рис. 3.4.2 Замена содержимого переменных

Вывожу в различных форматах значение регистра edx (рис. 3.4.3)

tssemenchenko@dk7n08 - lab09

Register group: general

eax	0x0	0
ecx	0x0	0
edx	0x0	0
ebx	0x0	0
esp	0xffffc330	0xffffc330
ebp	0x0	0x0
esi	0x0	0

0x804903c

add

BYTE PTR [eax],al

0x804903e

add

BYTE PTR [eax],al

0x8049040

add

BYTE PTR [eax],al

0x8049042

add

BYTE PTR [eax],al

0x8049044

add

BYTE PTR [eax],al

0x8049046

add

BYTE PTR [eax],al

native process 3045 (asm) In: _startL9PC:

(gdb) p/s \$eax

\$1 = 0

(gdb) p/t \$ecx

\$2 = 0

(gdb) p/t \$edx

\$3 = 0

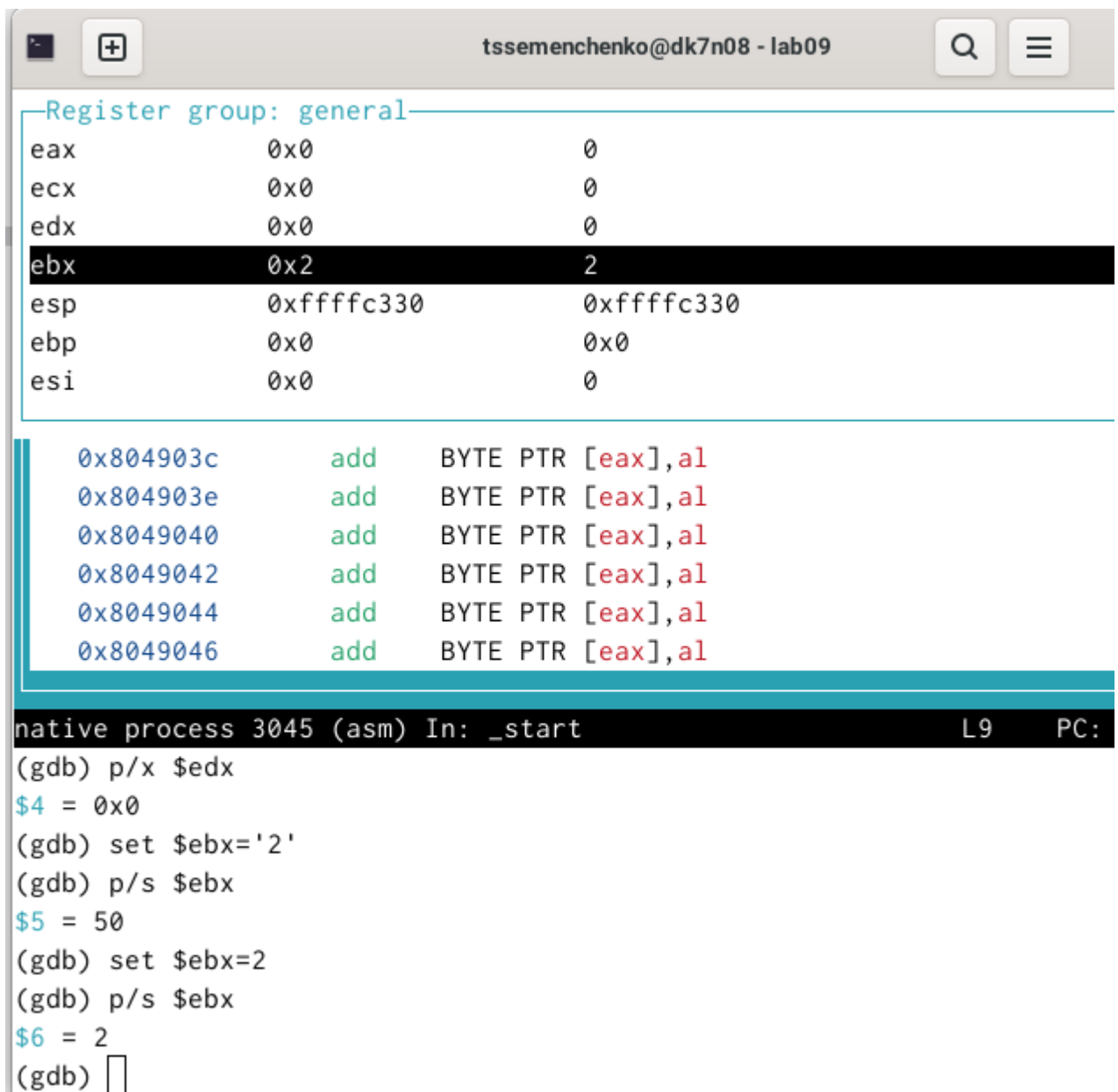
(gdb) p/x \$edx

\$4 = 0x0

(gdb)

Рис. 3.4.3 Различные значения регистра edx

С помощью команды set меняю содержимое регистра ebx (рис. 3.4.4)



The screenshot shows the GDB interface with the following content:

tssemenchenko@dk7n08 - lab09

Register group: general

Register	Value (Hex)	Value (Dec)
eax	0x0	0
ecx	0x0	0
edx	0x0	0
ebx	0x2	2
esp	0xffffc330	0xffffc330
ebp	0x0	0x0
esi	0x0	0

Assembly code:

```
0x804903c add BYTE PTR [eax],al
0x804903e add BYTE PTR [eax],al
0x8049040 add BYTE PTR [eax],al
0x8049042 add BYTE PTR [eax],al
0x8049044 add BYTE PTR [eax],al
0x8049046 add BYTE PTR [eax],al
```

native process 3045 (asm) In: _start L9 PC:

```
(gdb) p/x $edx
$4 = 0x0
(gdb) set $ebx='2'
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb) 
```

Рис. 3.4.4 Примеры использования команды set

3.5 Обработка аргументов командной строки в GDB

Скопировала файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab9-3.asm. Создала исполняемый файл. Загрузила исполняемый файл в отладчик, указав аргументы (рис. 3.5.1)

```
tssemenchenko@dk7n08 - lab09
Type nasm -h for help.
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ gdb --args lab9-3 аргумент1 аргумент 2 'аргумент 3'
Exception caught while booting Guile.
Error in function "load-thunk-from-memory":
missing DT_GUILLE_ENTRY
gdb: warning: Could not complete Guile gdb module initialization from:
/usr/share/gdb/guile/gdb/boot.scm.
Limited Guile support is available.
Suggest passing --data-directory=/path/to/gdb/data-directory.
GNU gdb (Gentoo 16.3 vanilla) 16.3
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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)

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.

(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/t/s/tssemenchenko/work/arch-pc/lab09/lab9-3 аргуме
нт1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:5
5          pop ecx
```

Рис. 3.5.1 Демонстрация работы программы

Установила точку останова перед первой инструкцией в программе и запустила ее (рис. 3.5.2)

```
(gdb) x/x $esp
0xffffc2f0: 0x00000005
```

Рис. 3.5.2 Установка точки останова

Запускаю программу в режиме отладки с указанием аргументов, указываю брейкпоинт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что

аргументы на вход программы закончились (рис. 3.5.3)

```
(gdb) x/x $esp
0xffffc2f0:      0x00000005
(gdb) x/x *(void**)(esp+4)
0xffffc556:      0x7366612f
(gdb) x/x *(void**)(esp+8)
0xffffc59f:      0x80d1b0d0
(gdb) x/x *(void**)(esp+12)
0xffffc5b1:      0x80d1b0d0
(gdb) x/x *(void**)(esp+16)
0xffffc5c2:      0xb0d00032
(gdb) x/x *(void**)(esp+20)
0xffffc5c4:      0x80d1b0d0
(gdb) x/x *(void**)(esp+24)
0x0:      Cannot access memory at address 0x0
(gdb) □
```

Рис. 3.5.3 Проверки работы стека

4 Задания для самостоятельной работы

Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 4.1)

```
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ ./lab9-4
Функция: f(x) = 5(2+x)
Результат: 0
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ □
```

Рис. 4.1 Измененная программа предыдущей лабораторной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 5(2+x) ",0
msg_res db "Результат: ",0

SECTION .bss
sum resb 1

SECTION .text
```

```

global _start
_start:
pop ecx
pop edx
sub ecx, 1
mov dword [sum],0
mov eax, msg_func
call sprintLF
jcxz _end
next_arg:
pop eax
call atoi
call _f_func
add [sum], eax
loop next_arg
_end:
mov eax, msg_res
call sprint
mov eax, [sum]
call iprintLF
call quit
_f_func:
push ebx
mov ebx, eax
add eax, 2
mov ecx, 5
mul ecx
pop ebx
ret

```

В листинге 9.3 приведена программа вычисления выражения $(3 + 2) * 4 + 5$. При запуске данная программа дает неверный результат. Создала файл lan9-5.asm,

вписала в него текст листинга 9.3 и проверила это (рис. 4.2)

```
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ touch lab9-5.asm
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ mc

tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 10
```

Рис. 4.2 Создание файла и его компиляция

Исправила программу и запустила ее (рис. 4.3)

```
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 25
tssemenchenko@dk7n08 ~/work/arch-pc/lab09 $ mc
```

Рис. 4.3 Запуск исправленной программы

Код программы:

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
mov ebx,3
mov eax,2
add ebx,eax
mov eax,ebx
mov ecx,4
mul ecx
mov ebx,eax
add ebx,5
mov edi,ebx
mov eax, div
call sprint
```

```
mov eax, edi  
call iprintLF  
call quit
```

5 Вывод

Приобрела навыки написания программ с использованием подпрограмм. Ознакомилась с методами отладки при помощи GDB и его основными возможностями.