

мехмет эфе кантоз группа НКАбд-04-23 Лабораторная работа № 9

1 Цель работы

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

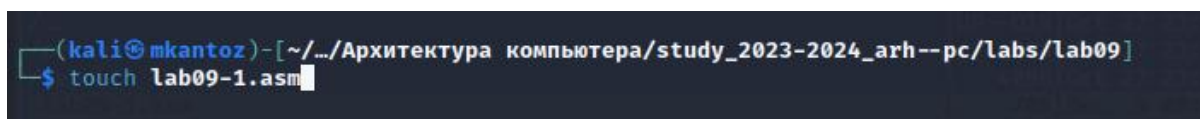
2 Задание

- 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. Задания для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Реализация подпрограмм в NASM.

- Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и создаю файл lab9-1.asm. (рис. [??]).



```
(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]  
$ touch lab09-1.asm
```

создание файлов для лабораторной работы

- Ввожу в файл lab09-1.asm текст программы с использованием подпрограммы из листинга 9.1. (рис. [??]).

```
mc [kali@mkantoz]:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09 (on mkantoz)
File Actions Edit View Help
GNU nano 7.2 /home/kali/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09/lab09-1.asm
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
res: 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 iprintf
call quit
;
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
```

ввод текста программы

- Создаю исполняемый файл и проверяю его работу. (рис. [??]).

```
(kali@mkantoz)-[~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ nasm -f elf lab09-1.asm
```

```
(kali@mkantoz)-[~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ld -m elf_i386 -o lab09-1 lab09-1.o
```

```
(kali@mkantoz)-[~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ./lab09-1
```

```
Введите x: 2
2x+7=11
```

запуск исполняемого файла

- Изменяю текст программы, добавив подпрограмму _subcalcul в подпрограмму _calcul для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$. (рис. [??]).

```
File Actions Edit View Help
GNU nano 7.2 /home/kali/work
%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
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx, 3
mul ebx
add eax, -1
ret
```

изменение текста программы

- Создаю исполняемый файл и проверяю его работу. (рис. [??]).

```
(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ nasm -f elf lab09-1.asm

(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ld -m elf_i386 -o lab09-1 lab09-1.o

.. not enough arguments

(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ./lab09-1
```

```
Введите x: 2
2x+7=17
```

запуск исполняемого файла

3.2 Отладка программ с помощью GDB.

- На этом шаге мы создали файл lab09-2.asm с текстом программы из листинга 9.2. (рис. [??]).

ввод текста программы

- Получаю исполняемый файл для работы с GDB с ключом '-g'. (рис. [??]).
-

```
(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ touch lab9-2.asm

(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ nasm -f elf lab9-2.asm

(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ld -m elf_i386 -o lab9-2 lab9-2.o
```

получение исполняемого файла

- Загружаю исполняемый файл в отладчик gdb.(рис. [??]).

```
(kali@mkantoz)-[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ gdb lab9-2
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 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 lab9-2...
(No debugging symbols found in lab9-2)
(gdb) █
```

загрузка исполняемого файла в отладчике

- Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run. (рис. [??]).

```
[Inferior 1 (process 677622) exited normally]
(gdb) break _start
Breakpoint 1 at 0x08049000
(gdb) run
Starting program: /home/kali/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09/lab9-2

Breakpoint 1, 0x08049000 in _start ()
(gdb) █
```

проверка работы файла с помощью команды run

- Для более подробного анализа программы устанавливаю брейкпоинт на метку _start и запускаю её.(рис. [??]).

- *включение режима псевдографики*

3.3 Добавление точек останова.

- Проверяю, что точка останова по имени метки `_start` установлена с помощью команды `info breakpoints` и устанавливаю еще одну точку останова по адресу инструкции `mov ebx,0x0`. Просматриваю информацию о всех установленных точках останова.(рис. [??]).

The screenshot shows a debugger window with two main panes. The top pane displays assembly instructions, and the bottom pane shows breakpoint information.

Assembly Instructions:

```

0x8049084  add  BYTE PTR [eax],a1
0x8049086  add  BYTE PTR [eax],a1
0x8049088  add  BYTE PTR [eax],a1
0x804908a  add  BYTE PTR [eax],a1
0x804908c  add  BYTE PTR [eax],a1
0x804908e  add  BYTE PTR [eax],a1
0x8049090  add  BYTE PTR [eax],a1
0x8049092  add  BYTE PTR [eax],a1
0x8049094  add  BYTE PTR [eax],a1
0x8049096  add  BYTE PTR [eax],a1
0x8049098  add  BYTE PTR [eax],a1
0x804909a  add  BYTE PTR [eax],a1
0x804909c  add  BYTE PTR [eax],a1

```

Breakpoint Information:

```

native process 4281 In: _start
Num  Type      Disp Enb Address  What
1    breakpoint keep y  0x08049000 <_start>
breakpoint already hit 1 time
(gdb) i b
Num  Type      Disp Enb Address  What
1    breakpoint keep y  0x08049000 <_start>
breakpoint already hit 1 time
(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)

```

установка точек останова

3.4 Работа с данными программы в GDB.

- Выполняю 5 инструкций с помощью команды `stepi` и слежу за изменением значений регистров. (рис. [??]).


```
File Actions Edit View Help
kal@mkantoz: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ark-pc/labs/lab09 (on mkantoz)

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
0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80
0x8049038 add BYTE PTR [eax],al
0x8049039 add BYTE PTR [eax],al
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
0x8049048 add BYTE PTR [eax],al
0x804904a add BYTE PTR [eax],al
0x804904c add BYTE PTR [eax],al
0x804904e add BYTE PTR [eax],al
0x8049050 add BYTE PTR [eax],al
0x8049052 add BYTE PTR [eax],al
0x8049054 add BYTE PTR [eax],al

native process 678021 In: _start
(gdb) cd
Working directory /home/kali.
(gdb) layout asm
(gdb) |
```

```
[ Register Values Unavailable ]

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
0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80
0x8049038 add BYTE PTR [eax],al

native process 678021 In: _start
(gdb) layout regs
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 <_start>
breakpoint already hit 1 time
(gdb) b '*+%/)=?0~0'
Function '*+%/)=?0~0' not defined.
(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) |
```

```

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

native process 678021 In: _start
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffcef0 0xffffcef0
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
--Type <RET> for more, q to quit, c to continue without paging--

```

до использования команды *stepi*

(рис. [??]).

```

--Register group: general
eax 0x8 8 ecx 0x804a000 134520832 edx 0x8 8
ebx 0x1 1 esp 0xffffc300 0xffffc300 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
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
b+ 0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80

native process 4281 In: _start
esp 0xffffc300 0xffffc300
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
--Type <RET> for more, q to quit, c to continue without paging--ccsfs 0x0 0
(gdb) si 5

```

после использования команды *stepi*

- Просматриваю значение переменной msg1 по имени с помощью команды x/1sb &msg1 и значение переменной msg2 по ее адресу.(рис. [??]).

```

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

native process 678021 In: _start
esp    0xffffcef0    0xffffcef0
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
--Type <RET> for more, q to quit, c to continue without paging--fs    0x0    0
gs     0x0           0
(gdb) x/1sb &msg1
0x804a000: "Hello, "
(gdb)

```

просмотр значений переменных

- С помощью команды set изменяю первый символ переменной msg1 и заменяю первый символ в переменной msg2.(рис. [??]).

```

native process 678021 In: _start
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000: "hello, "
(gdb) set {char}&msg2='b'
(gdb) x/1sb &msg2
0x804a008: "borld!\n"
(gdb)

```

использование команды set

- Вывожу в шестнадцатеричном формате, в двоичном формате и в символьном виде соответственно значение регистра edx с помощью команды print p/F \$val.(рис. [??]).

```

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 678021 In: _start
(gdb) set {char} &msg1='h'
(gdb) x/1sb &msg1
0x804a000: "hello, "
(gdb) set {char} &msg2='b'
(gdb) x/1sb &msg2
0x804a008: "borld!\n"
(gdb) p/x $edx
$1 = 0x0
(gdb) p/t $edx
$2 = 0
(gdb) p/c $edx
$3 = 0 '\000'
(gdb)

```

вывод значения регистра

- С помощью команды set изменяю значение регистра ebx в соответствии с заданием. (рис. [??]).

```

Register group: general
eax      0x8      8      ecx      0x804a000      134520832      edx      0x8      8
ebx      0x2      2      esp      0xffffc300      0xffffc300      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
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80

native process 4281 In: _start
(gdb) x/1sb &msg2
0x804a008: "borld!\n"
(gdb) p/x $edx
$1 = 0x8
(gdb) p/t $edx
$2 = 1000
(gdb) p/c $edx
$3 = 8 '\b'
(gdb) set $ebx=2
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb)

```

использование команды set для изменения значения регистра

- Разница вывода команд p/s \$ebx отличается тем, что в первом случае мы переводим символ в его строковый вид, а во втором случае число в строковом виде не изменяется.

- Завершаю выполнение программы с помощью команды continue и выхожу из GDB с помощью команды quit.(рис. [??]).

```

File Actions Edit View Help
eax 0x0 0 ecx 0x0 0 edx 0x0 0
esi 0x1 1 edi 0x804a008 134520840 edx 0x7 7
eflags 0x202 [ IF ] cs 0x23 35 eip 0x8049000 0x8049000 <_start>
ds 0x2b 43 es 0x2b 43 fs 0x0 0
gs 0x0 0

0x8049005 <_start+5> mov edi,0x1
0x804900f <_start+15> mov edi,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov edi,0x1
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edi,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov edi,0x1
0x8049031 <_start+49> mov edi,0x8
B> 0x8049031 <_start+49> mov ebx,0x0
0x804903d add BYTE PTR [eax],al
native process 678021 in: _start
(gdb) p/t $edx
$5 = 0x32
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb) c
Continuing.
hello, world!
Breakpoint 2, 0x8049031 in _start ()
(gdb) q
A debugging session is active.

Inferior 1 [process 678021] will be killed.

Quit anyway? (y or n)

```

завершение работы

3.5 Обработка аргументов командной строки в GDB.

- Копирую файл lab8-2.asm с программой из листинга 8.2 в файл с именем lab09-3.asm и создаю исполняемый файл. (рис. [??]).

```

(kali@mkantoz)-[~/../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ nasm -f elf lab9-3.asm

(kali@mkantoz)-[~/../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ld -m elf_i386 -o lab9-3 lab9-3.o

(kali@mkantoz)-[~/../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ gdb --args lab09-3 аргумент 1 аргумент2 'аргумент3'
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 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"...
lab09-3: No such file or directory.
(gdb) b _start

```

загрузка исполняемого файла в отладчике

- Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее.(рис. [??]).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/s/e/serazanacua/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx
(gdb) |
```

установка точек останова

- Посматриваю вершину стека и позиции стека по их адресам. (рис. [??]).

```
(gdb) x/x $esp
0xffffc2b0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffc54f: "/afs/.dk.sci.pfu.edu.ru/home/s/e/serazanacua/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc597: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc5a9: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc5ba: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc5bc: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb) |
```

просмотр значений и введение в стек

3.6 Задания для самостоятельной работы.

- - 1) Преобразовываю программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.
 - Запускаю код и проверяю, что она работает корректно. (рис. [??]).

```
(kali@mkantoz)~[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ touch test1.asm

(kali@mkantoz)~[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ nasm -f elf test1.asm

(kali@mkantoz)~[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ld -m elf_i386 -o test1 test1.o
```

```
(kali@mkantoz)~[~/.../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ./test1 2 5 7
```

Функция : $f(x) = 30x - 11$
 Результат : 387

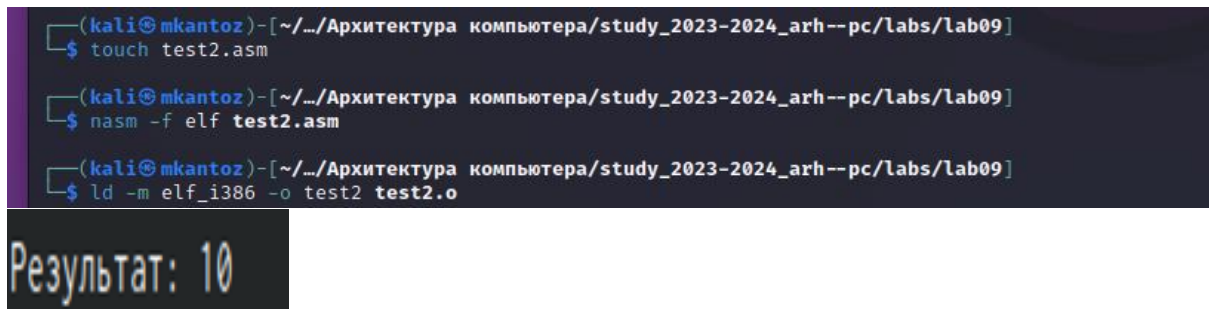
запуск программы

- - 2) Ввожу в файл task1.asm текст программы из листинга 9.3
 - При корректной работе программы должно выводиться "25". Создаю исполняемый файл и запускаю его.(рис. [??]).

```
(kali@mkantoz)~[~/../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ touch test2.asm

(kali@mkantoz)~[~/../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ nasm -f elf test2.asm

(kali@mkantoz)~[~/../Архитектура компьютера/study_2023-2024_arh--pc/labs/lab09]
$ ld -m elf_i386 -o test2 test2.o
```



запуск программы

4 Выводы

- Во время выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм и ознакомилась с методами отладки при помощи GDB и его основными возможностями.

