

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 09

<<ПОНЯТИЕ ПОДПРОГРАММЫ. ОТЛАДЧИК GDB >>

дисциплина: *Архитектура компьютера*

Студент: ИБРАХИМ ХИССЕИН ГАНА

Группа: НПИбд 01-25

МОСКВА

2025г.

1. Цель работы

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

2. Реализация подпрограмм в NASM

2.1 Программа lab09-1.asm (вычисление $2x+7$)

Исходная программа:

```
```assembly
%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:
 mov eax, msg
 call sprint
 mov ecx, x
 mov edx, 80
 call sread
 mov eax,x
 call atoi
 call _calcul
 mov eax,result
 call sprint
 mov eax,[res]
 call iprintLF
```

```
call quit
```

```
_calcul:
```

```
 mov ebx,2
 mul ebx
 add eax,7
 mov [res],eax
 ret
```

**Создание файлов:**

```
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ ls -la lab09-1.asm in
_out.asm
-rw-rw-r-- 1 ibrahim ibrahim 3799 déc. 6 03:25 in_out.asm
-rw-rw-r-- 1 ibrahim ibrahim 742 déc. 6 03:45 lab09-1.asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$
```

**Тестирование оригинальной программы ( $x=5 \rightarrow 17$ ):**

```
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nano lab09-1.asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nasm -f elf lab09-1.a
sm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab
09-1 lab09-1.o
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ echo "2" | ./lab09-1
Введите x: 2x+7=17
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ echo "4" | ./lab09-1
Введите x: 2x+7=29
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ echo "0" | ./lab09-1
Введите x: 2x+7=5
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$
```

## **2.2 Модификация с подпрограммой subcalcul Измененный код (добавлена subcalcul для $g(x)=3x-1$ ):**

```
_subcalcul:
```

```
 mov ebx, 3
 mul ebx ; eax = 3x
 sub eax, 1 ; eax = 3x - 1
```

```

 ret

_calcul:
 call _subcalcul ; eax = g(x)
 mov ebx, 2
 mul ebx ; eax = 2 * g(x)
 add eax, 7 ; eax = 2*g(x) + 7
 mov [res], eax
 ret

```

**Код после исправления синтаксических ошибок:**

```

 mov eax,[res]
 call iprintLF

 call quit

; --- programme _subcalcul (g(x) = 3x - 1) ---
_subcalcul:
 mov ebx, 3
 mul ebx ; eax = 3*x
 sub eax, 1 ; eax = 3*x - 1
 ret

; --- programme _calcul (f(g(x))) ---
_calcul:
 call _subcalcul ; eax = g(x)
 mov ebx, 2
 mul ebx ; eax = 2 * g(x)
 add eax, 7 ; eax = 2*g(x) + 7
 mov [res], eax
 ret

```

ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09\$ █

**Компиляция исправленной программы:**

```

ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ █

```

**Тестирование модифицированной программы:**

$x=2 \rightarrow f(g(2)) = 2(32-1)+7 = 2*5+7 = 17$

$x=4 \rightarrow f(g(4)) = 2(34-1)+7 = 2*11+7 = 29$

$x=0 \rightarrow f(g(0)) = 2(30-1)+7 = 2*(-1)+7 = 5$

```
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nano lab09-1.asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ echo "2" | ./lab09-1
Введите x: 2x+7=17
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ echo "4" | ./lab09-1
Введите x: 2x+7=29
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ echo "0" | ./lab09-1
Введите x: 2x+7=5
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$
```

### **3. Отладка с помощью GDB 3.1 Программа Hello World (lab09-2.asm)**

#### **Исходный код программы:**

SECTION .data

```
msg1: db "Hello,",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
```

SECTION .text

```
global _start
```

```
_start:
```

```
 mov eax, 4
 mov ebx, 1
 mov ecx, msg1
 mov edx, msg1Len
 int 0x80
 mov eax, 4
 mov ebx, 1
 mov ecx, msg2
 mov edx, msg2Len
 int 0x80
 mov eax, 1
 mov ebx, 0
 int 0x80
```

**Создание файла:**

```

ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ cat lab09-2.asm
SECTION .data
 msg1: db "Hello, ",0x0
 msg1Len: equ $ - msg1

 msg2: db "world!",0xa
 msg2Len: equ $ - msg2

SECTION .text
 global _start

_start:
 mov eax, 4
 mov ebx, 1
 mov ecx, msg1
 mov edx, msg1Len
 int 0x80

 mov eax, 4
 mov ebx, 1
 mov ecx, msg2
 mov edx, msg2Len
 int 0x80

 mov eax, 1
 mov ebx, 0
 int 0x80

```

Компиляция с отладочной информацией (-g):

```

ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nasm -f elf -g -l lab
09-2.lst lab09-2.asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab
09-2 lab09-2.o
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ ls -la lab09-2 lab09-
2.o lab09-2.lst
-rwxrwxr-x 1 ibrahim ibrahim 9152 déc. 6 03:51 lab09-2
-rw-rw-r-- 1 ibrahim ibrahim 1421 déc. 6 03:51 lab09-2.lst
-rw-rw-r-- 1 ibrahim ibrahim 1616 déc. 6 03:51 lab09-2.o
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$

```

Запуск программы в GDB:

```

(gdb) run
Starting program: /home/ibrahim/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 6172) exited normally]
(gdb)

```

### 3.2 Основные возможности GDB

Установка точки останова на `_start`:

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 12.
(gdb) run
Starting program: /home/ibrahim/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:12
12 mov eax, 4
(gdb)

```

Дизассемблирование (синтаксис AT&T и Intel):

```

Breakpoint 1, _start () at lab09-2.asm:12
12 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.
(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)

```

### Графический режим GDB (layout asm + layout regs):

The screenshot shows the GDB graphical interface with two panes. The top pane displays the general register values:

Register	Value	Description
eax	0x0	0
ecx	0x0	0
edx	0x0	0
ebx	0x0	0
esp	0xfffffce0	0xfffffce0
ebp	0x0	0x0

The bottom pane shows the assembly code for the \_start function:

```

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

```

At the bottom, the status bar indicates: native process 6189 (asm) In: \_start L12 PC: 0x8049000.

### Установка точки останова по адресу инструкции:

```
(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) 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) layout asm
(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) break *0x08049031
Breakpoint 2 at 0x08049031: file lab09-2.asm, line 25.
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:12
2 breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab09-2.asm:25
(gdb)
```

### Просмотр регистров (info registers):

```
(gdb) info registers
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xfffffceb0 0xfffffceb0
ebp 0x0 0x0
esi 0x0 0
edi 0x0 0
eip 0x80490000 0x8049000 <_start>
eflags 0x202 [IF]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x0 0
<0> 0x0 0
<1> 0x0 0
<2> 0x0 0
<3> 0x0 0
<4> 0x0 0
<5> 0x0 0
<6> 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--
```

### Просмотр содержимого памяти (команда x):

```
(gdb) info registers
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xfffffceb0 0xfffffceb0
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
<0> 0x0 0
<1> 0x0 0
<2> 0x0 0
<3> 0x0 0
<4> 0x0 0
<5> 0x0 0
<6> 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--
```

### Изменение содержимого памяти:

```
[Quitting]
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) (gdb) x/1sb &msg2
Undefined command: "". Try "help".
(gdb) x/1sb &msg2
0x804a008 <msg2>: "world!\n\034"
(gdb)
```

### Работа с регистрами (разные форматы вывода):

```
(gdb) x/1sb &msg2
0x804a008 <msg2>: "World!\n\034"
(gdb) print /x $edx
$1 = 0x0
(gdb) print /t $edx
$2 = 0
(gdb) print /c $edx
$3 = 0 '\000'
(gdb) set $ebx=2
(gdb) print /s $ebx
$4 = 2
(gdb)
```

### 3.3 Аргументы командной строки (lab09-3.asm)

Создание файла lab09-3.asm:

```
(gdb) x/1$D &msgz
0x804a008 <msg2>: "World! \n\034"
(gdb) print /x $edx
$1 = 0x0
(gdb) print /t $edx
$2 = 0
(gdb) print /c $edx
$3 = 0 '\000'
(gdb) set $ebx=2
(gdb) print /s $ebx
$4 = 2
(gdb)
```

Компиляция программы:

```
gcc -o lab09-3 lab09-3.c
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$./lab09-3
World!
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$
```

Запуск GDB с аргументами:

```
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ gdb --args lab09-3 argument1 argument2 'argument 3'
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
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-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)
```

Анализ расположения аргументов в стеке:

```
breakpoint 1 at 0x401000: file lab09-3.asm, line 7.
(gdb) run
Starting program: /home/ibrahim/work/arch-pc/lab09/lab09-3 argument1 argument2 a
rgument\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
 <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab09-3.asm:7
7 pop ecx
(gdb) x/x $esp
0xfffffc90: 0x00000004
(gdb) x/s *(void**)($esp + 4)
0xfffffd080: "/home/ibrahim/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xfffffd0a9: "argument1"
(gdb) x/s *(void**)($esp + 12)
0xfffffd0b3: "argument2"
(gdb) x/s *(void**)($esp + 16)
0xfffffd0bd: "argument 3"
(gdb)
```

x/x \$esp → 0x00000004 (4 аргумента: программа + 3 аргумента)

x/s \*(void\*\*)\$esp + 4 → “/home/.../lab09-3” (имя программы)

x/s \*(void\*\*)\$esp + 8 → “argument1”

x/s \*(void\*\*)\$esp + 12 → “argument2”

x/s \*(void\*\*)\$esp + 16 → “argument 3”

## **4. Самостоятельная работа**

### **4.1 Программа с подпрограммой (lab09-4.asm)**

Преобразование программы из лабораторной работы №9 с использованием подпрограммы:

```
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ cat lab09-4.asm
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ', 0h

SECTION .bss
N resb 10

SECTION .text
global _start
_start:
 ; --- Enter ---
 mov eax, msg1
 call sprint

 mov ecx, N
 mov edx, 10
 call sread

 mov eax, N
 call atoi
 mov [N], eax

 call _print_sequence

 call quit

._print_sequence:
 mov ecx, [N]

._print_loop:
 push ecx
 mov eax, ecx
 call iprintLF
 pop ecx
 loop .print_loop

 ret
```

Тестирование программы (ввод N=5):

```
Введите N: 5
5
4
3
2
1
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$
```

### **4.2 Отладка и исправление программы (lab09-5.asm)**

Оригинальная программа с ошибкой (вычисление  $(3+2)*4+5$ ):

```
mov ebx,3
mov eax,2
add ebx,eax ; ebx = 5
mov ecx,4
mul ecx ; ОШИБКА: eax=2, поэтому 2*4=8
add ebx,5 ; ebx = 10
```

## **Отладка в GDB для обнаружения ошибки:**

```
eax 0x0 0 ecx 0x0 0 edx 0x0 0
ebx 0x0 0 esp 0xfffffce0 0x00000000 ebp 0x0 0
esi 0x0 0 0x10 0x0 0x00490e8 <_start>
eflags 0x202 [IF] cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43 fs 0x0 0
gs 0x0 0 k0 0x0 0 k1 0x0 0
k2 0x0 0 k3 0x0 0 k4 0x0 0
k5 0x0 0 k6 0x0 0 k7 0x0 0

*0x00490e8 < start> mov $0x3,ebx
0x00490e00 < start+5> mov $0x0,ebx
0x00490f1 < start+10> add ebx,ebx
0x00490f12 < start+12> mov $0x0,ebx
0x00490f17 < start+17> add ebx,ebx
0x00490f19 < start+19> add ebx,ebx
0x00490f1a < start+20> add ebx,ebx
0x00490f1c < start+22> mov ebx,edi
0x00490f1d < start+24> mov $0x804a000,ecx
0x00490f1f < start+29> call 0x004900f <sprint>
0x00490f20 < start+34> mov edi,ecx
0x00490f20 < start+36> call 0x0049006 <printfLF>
0x00490f21 < start+41> call 0x00490db <quit>
0x0049116 add $0x1,(bx)
0x0049118 add $0x1,(bx)
0x004911a add $0x1,(bx)
0x004911c add $0x1,(bx)
0x004911e add $0x1,(bx)
```

### **Исправленная программа:**

```
mov ebx,3
mov eax,2
add ebx,ebx ; ebx = 5
mov ecx,4 mov eax,ebx ; ИСПРАВЛЕНИЕ: eax = ebx = 5
mul ecx ; eax = 5*4 = 20
add eax,5 ; eax = 25
```

```

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-5...
(gdb) break _start
Breakpoint 1 at 0x80490e8: file lab09-5.asm, line 11.
(gdb) run
Starting program: /home/ibrahim/work/arch-pc/lab09/lab09-5

This GDB supports auto-downloading debuginfo from the following URLs:
 <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) layout asm
Please answer y or [n].
Enable debuginfod for this session? (y or [n]) layout regs
Please answer y or [n].
Enable debuginfod for this session? (y or [n]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab09-5.asm:11
11 mov ebx,3
(gdb) layout asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nano lab09-5.asm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ █

```

Тестирование исправленной программы:

```

ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ nasm -f elf lab09-5.a
sm
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab
09-5 lab09-5.o
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$./lab09-5
Результат: 25
ibrahim@ibrahim-IdeaPad-5-Pro-14ITL6:~/work/arch-pc/lab09$ █

```

**Результат:** 25 (правильно:  $(3+2)4+5 = 54+5 = 20+5 = 25$ )

## 5. Выводы

В ходе лабораторной работы были успешно выполнены все поставленные задачи:

Освоены подпрограммы в **NASM**: созданы и использованы подпрограммы `_calcul` и `_subcalcul` для вычисления сложных выражений.

Приобретены навыки отладки с **GDB**:

Установка точек останова (по имени метки и по адресу)

Пошаговое выполнение (**stepi**, **nexti**)

Просмотр и изменение регистров и памяти

Дизассемблирование в разных синтаксисах (**AT&T/Intel**)

Исследованы механизмы передачи данных:

Работа со стеком при вызове подпрограмм

Анализ аргументов командной строки в стеке

Найден и исправлен логическая ошибка в программе вычисления арифметического выражения с использованием отладчика **GDB**.

Преобразована программа из предыдущей лабораторной работы для использования подпрограмм.

Цель работы достигнута полностью: приобретены практические навыки написания программ с подпрограммами и использования отладчика **GDB** для поиска и исправления ошибок.

## **6. Ответы на вопросы для самопроверки**

**1. Какие языковые средства используются в ассемблере для оформления и активизации подпрограмм?**

- Инструкции call (вызов) и ret (возврат), использование стека для сохранения адреса возврата.

**2. Объясните механизм вызова подпрограмм.**

- call сохраняет адрес следующей инструкции в стеке и передает управление на адрес подпрограммы. ret извлекает адрес возврата из стека.

**3. Как используется стек для обеспечения взаимодействия между вызывающей и вызываемой процедурами?**

- Стек используется для сохранения адреса возврата, передачи параметров и сохранения значений регистров.

**4. Каково назначение операнда в команде ret?**

- В NASM ret без операнда просто возвращает управление. С операндом ret N дополнительно очищает N байт из стека.

**5. Для чего нужен отладчик?**

- Для анализа выполнения программы, поиска ошибок, просмотра состояния регистров и памяти.

**6. Объясните назначение отладочной информации и как нужно компилировать программу, чтобы в ней присутствовала отладочная информация.**

- Отладочная информация связывает машинный код с исходным текстом.  
Компиляция с ключом -g.

**7. Расшифруйте и объясните следующие термины: breakpoint, watchpoint, checkpoint, catchpoint и call stack.**

- Breakpoint: точка останова (остановка на конкретной строке/адресе)
- Watchpoint: точка наблюдения (остановка при изменении переменной)
- Call stack: стек вызовов (история вызовов подпрограмм)

**8. Назовите основные команды отладчика gdb и как они могут быть использованы для отладки программ.**

- break: установка точки останова
- run: запуск программы
- stepi/nexti: пошаговое выполнение
- info registers: просмотр регистров
- x: просмотр памяти
- print: вывод значений