

# **Отчёт по лабораторной работе 9**

**Архитектура компьютеров**

Акмухаммедов Айдын

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	27

## Список иллюстраций

2.1	Программа в файле lab9-1.asm . . . . .	7
2.2	Запуск программы lab9-1.asm . . . . .	7
2.3	Программа в файле lab9-1.asm . . . . .	8
2.4	Запуск программы lab9-1.asm . . . . .	9
2.5	Программа в файле lab9-2.asm . . . . .	10
2.6	Запуск программы lab9-2.asm в отладчике . . . . .	11
2.7	Дизассимилированный код . . . . .	12
2.8	Дизассимилированный код в режиме интел . . . . .	13
2.9	Точка остановки . . . . .	14
2.10	Изменение регистров . . . . .	15
2.11	Изменение регистров . . . . .	16
2.12	Изменение значения переменной . . . . .	17
2.13	Вывод значения регистра . . . . .	18
2.14	Вывод значения регистра . . . . .	19
2.15	Вывод значения регистра . . . . .	20
2.16	Программа в файле lab9-4.asm . . . . .	21
2.17	Запуск программы lab9-4.asm . . . . .	22
2.18	Код с ошибкой . . . . .	23
2.19	Отладка . . . . .	24
2.20	Код исправлен . . . . .	25
2.21	Проверка работы . . . . .	26

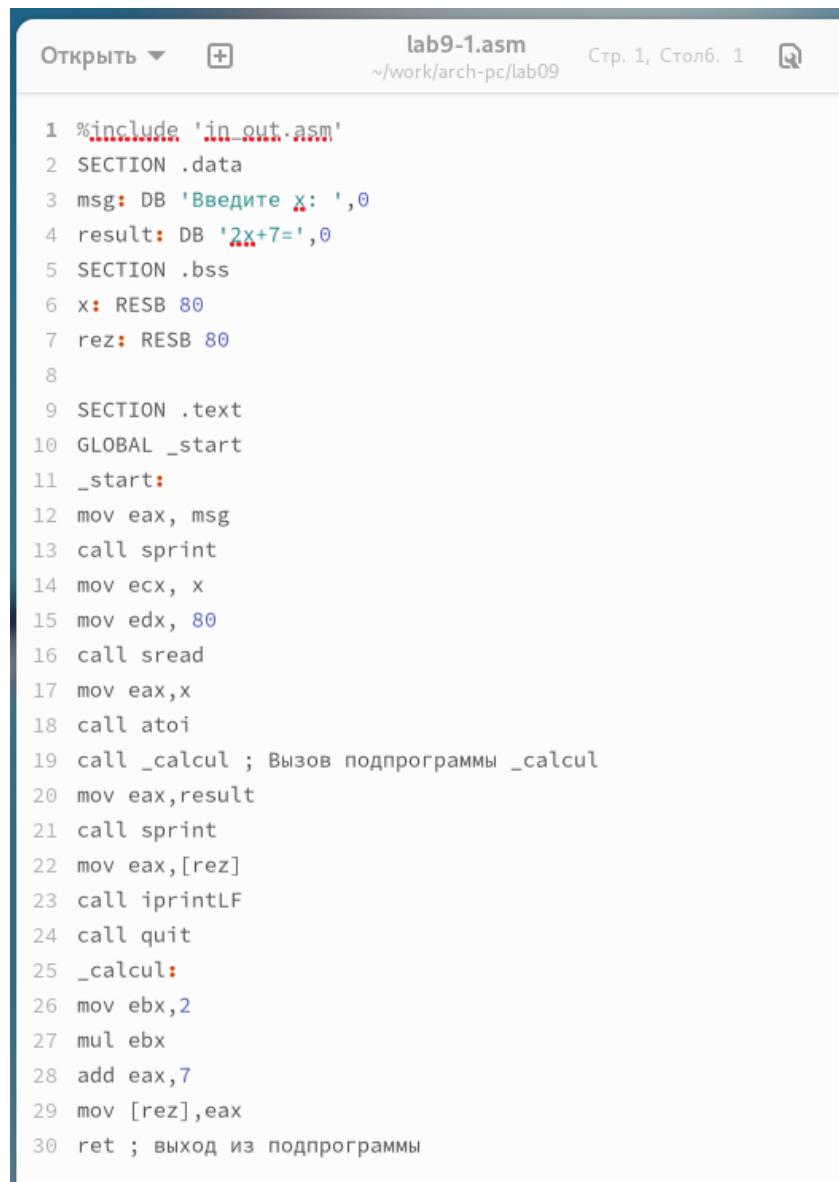
## Список таблиц

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

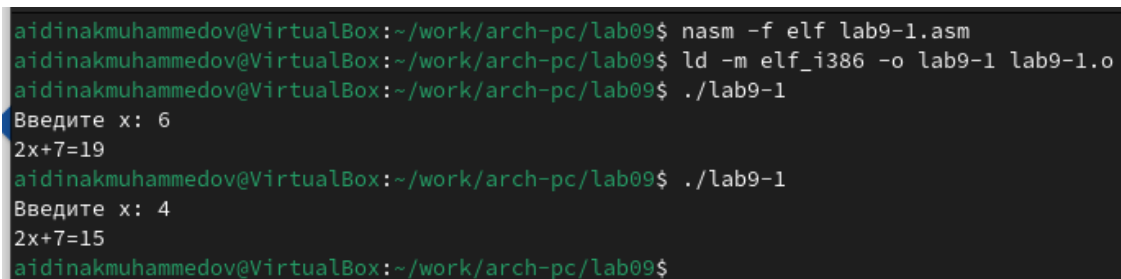
## 2 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы № 9, перешел в него и создал файл lab9-1.asm.
2. В качестве примера рассмотрим программу вычисления арифметического выражения  $f(x) = 2x + 7$  с помощью подпрограммы calcul. В данном примере  $x$  вводится с клавиатуры, а само выражение вычисляется в подпрограмме.



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

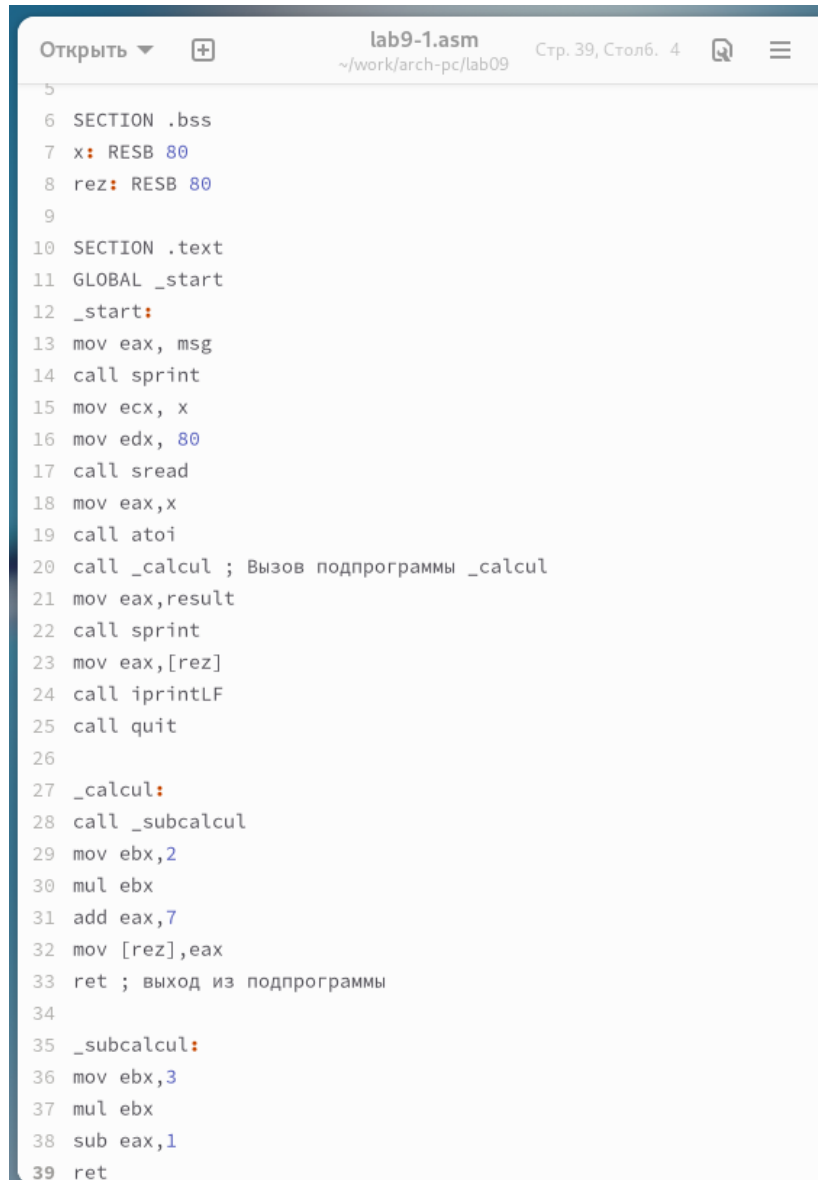
Рис. 2.1: Программа в файле lab9-1.asm



```
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2x+7=19
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 4
2x+7=15
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.2: Запуск программы lab9-1.asm

3. Изменил текст программы, добавив подпрограмму `subcalcul` в подпрограмму `calcul`, для вычисления выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ .



```
5
6 SECTION .bss
7 x: RESB 80
8 rez: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x
19 call atoi
20 call _calcul ; Вызов подпрограммы _calcul
21 mov eax, result
22 call sprint
23 mov eax, [rez]
24 call iprintLF
25 call quit
26
27 _calcul:
28 call _subcalcul
29 mov ebx, 2
30 mul ebx
31 add eax, 7
32 mov [rez], eax
33 ret ; выход из подпрограммы
34
35 _subcalcul:
36 mov ebx, 3
37 mul ebx
38 sub eax, 1
39 ret
```

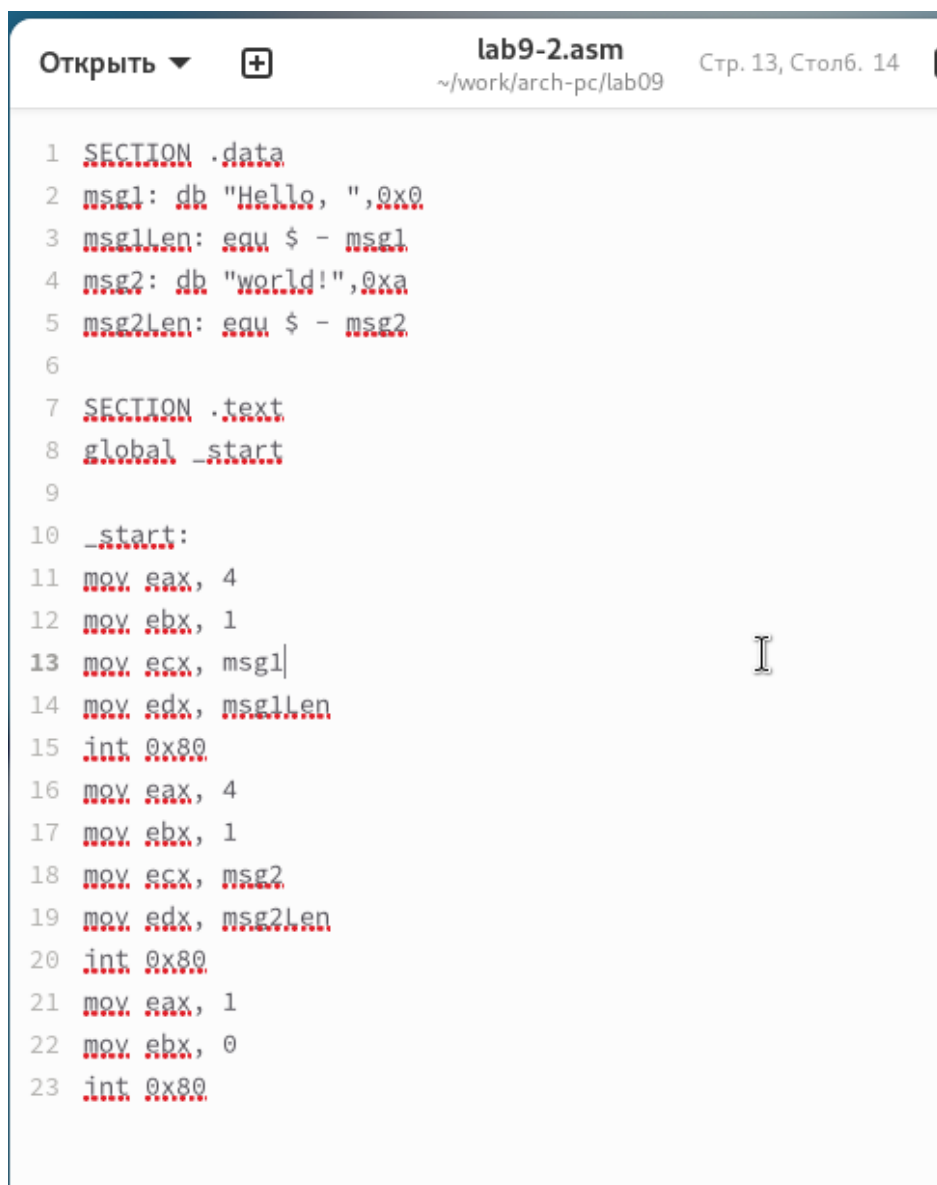
Рис. 2.3: Программа в файле lab9-1.asm



```
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$  
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm  
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o  
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 6  
2(3x-1)+7=41  
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 4  
2(3x-1)+7=29  
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.4: Запуск программы lab9-1.asm

4. Создал файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!).



```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msgllen: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2len: equ $ - msg2
6
7 SECTION .text
8 global _start
9
10 _start:
11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msgllen
15 int 0x80
16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2len
20 int 0x80
21 mov eax, 1
22 mov ebx, 0
23 int 0x80
```

Рис. 2.5: Программа в файле lab9-2.asm

Получил исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’.

Загрузил исполняемый файл в отладчик gdb. Проверил работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r).

```

aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.1-1.fc39
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-redhat-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...
(gdb) r
Starting program: /home/aidinakmuhammedov/work/arch-pc/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4232) exited normally]
(gdb)

```

Рис. 2.6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы установите брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запустите её. Посмотрите дисассимилированный код программы.

```
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Reading symbols from lab9-2...
(gdb) r
Starting program: /home/aidinakmuhammedov/work/arch-pc/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4232) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/aidinakmuhammedov/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:11
11      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) 
```

Рис. 2.7: Дизассимилированный код

```
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
11      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) 
```

Рис. 2.8: Дизассимилированный код в режиме интел

На предыдущих шагах была установлена точка остановки по имени метки (`_start`). Проверил это с помощью команды `info breakpoints` (кратко `i b`). Установил еще одну точку остановки по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определил адрес предпоследней инструкции (`mov ebx,0x0`) и установил точку.

The screenshot shows a GDB terminal window with the title bar "aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into two main sections. The top section, titled "Register group: general", lists the values of various registers: `eax` (0x0), `ecx` (0x0), `edx` (0x0), `ebx` (0x0), `esp` (0xffffd0a0), `ebp` (0x0), `esi` (0x0), `edi` (0x0), and `eip` (0x8049000). The bottom section displays assembly instructions being stepped through, starting from `0x8049000 <_start>` with `mov eax,0x4` and continuing through several `mov` and `int` instructions. Below the assembly view, a status bar indicates "native process 4237 (asm) In: \_start" and "L11 PC: 0x8049000". The GDB command history shows `(gdb) layout regs`, `(gdb) b *0x8049031`, and `(gdb) i b`. A table of breakpoints is displayed, showing two breakpoints: one at `0x08049000` (hit 1 time) and another at `0x08049031`.

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>

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

native process 4237 (asm) In: _start L11 PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint     keep y   0x08049000 lab9-2.asm:11
       breakpoint already hit 1 time
2      breakpoint     keep y   0x08049031 lab9-2.asm:22
(gdb)
```

Рис. 2.9: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды `stepi` (или `si`) и проследил за изменением значений регистров.

```
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>

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

native process 4237 (asm) In: _start L12 PC: 0x8049005
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb) 
```

Рис. 2.10: Изменение регистров

The screenshot shows a GDB terminal window with the title bar "aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of general-purpose registers: 

eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0a0	0xffffd0a0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

. The middle section displays assembly instructions with their addresses and operands: 

```
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
```

. The bottom section shows the state of segment registers: 

```
native process 4237 (asm) In: _start          L16    PC: 0x8049016
--Type <RET> for more, q to quit, c to continue without paging--
ss          0x2b          43
ds          0x2b          43
es          0x2b          43
fs          0x0           0
gs          0x0           0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.11: Изменение регистров

Посмотрел значение переменной msg1 по имени. Посмотрел значение переменной msg2 по адресу.

Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Изменил первый символ переменной msg1.



The screenshot shows a GDB terminal window titled "aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into two main sections. The top section, titled "Register group: general", displays the current values of general-purpose registers: 

eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0a0	0xffffd0a0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

. The bottom section displays assembly code for the current instruction at address 0x8049016: 

```
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
```

. Below the assembly code, the status bar shows "native process 4237 (asm) In: \_start L16 PC: 0x8049016". The bottom section shows the GDB command prompt with the following commands and output: 

```
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>:    "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:    "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:    "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:    "Lorld!\n\034"
(gdb) 
```

Рис. 2.12: Изменение значения переменной

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.

The screenshot shows a GDB terminal window with the title bar "aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of general-purpose registers: 

Register	Value (hex)	Value (decimal)
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0a0	0xffffd0a0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

. The middle section displays assembly code with addresses and instructions: 

```
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
```

. The bottom section shows the native process 4237 (asm) in the \_start function, with a list of variables and their values: 

```
$2 = 1000
(gdb) p/s $ecx
$3 = 134520832
(gdb) p/x $ecx
$4 = 0x804a000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx

The screenshot shows a GDB terminal window with the title bar "aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", lists the values of general-purpose registers: `eax` (0x8), `ecx` (0x804a000), `edx` (0x8), `ebx` (0x2), `esp` (0xffffd0a0), `ebp` (0x0), `esi` (0x0), `edi` (0x0), and `eip` (0x8049016). The middle section displays assembly instructions with their addresses and disassembled forms, such as `0x8049000 <_start> mov eax,0x4`. The bottom section shows the GDB command prompt with a list of commands entered: `$5 = 8`, `(gdb) p/t $edx`, `$6 = 1000`, `(gdb) p/x $edx`, `$7 = 0x8`, `(gdb) set $ebx='2'`, `(gdb) p/s $ebx`, `$8 = 50`, `(gdb) set $ebx=2`, `(gdb) p/s $ebx`, `$9 = 2`, and `(gdb)`. The status bar at the bottom indicates "native process 4237 (asm) In: \_start" and "PC: 0x8049016".

```
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>

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

native process 4237 (asm) In: _start          L16    PC: 0x8049016
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb)
```

Рис. 2.14: Вывод значения регистра

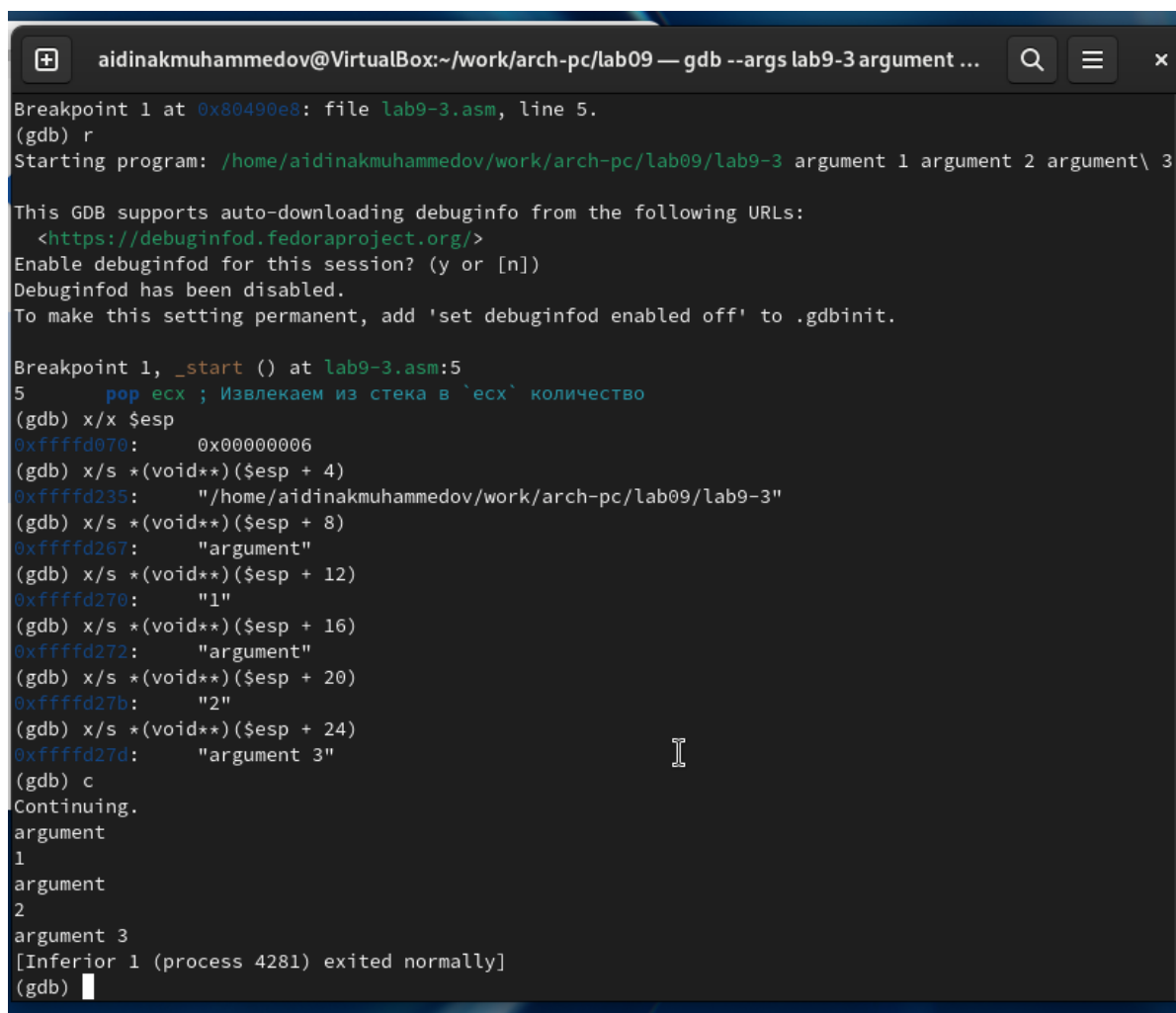
5. Скопировал файл `lab8-2.asm`, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создал исполняемый файл. Для загрузки в `gdb` программы с аргументами необходимо использовать ключ `-args`. Загрузил исполняемый файл в отладчик, указав аргументы.

Для начала установил точку останова перед первой инструкцией в программе и запустил ее.

Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы `lab9-3` и

непосредственно аргументы: аргумент1, аргумент, 2 и 'аргумент 3'.

Посмотрел остальные позиции стека – по адресу [esp+4] располагается адрес в памяти где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12] – второго и т.д.



```
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb --args lab9-3 argument ...
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/aidinakmuhammedov/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

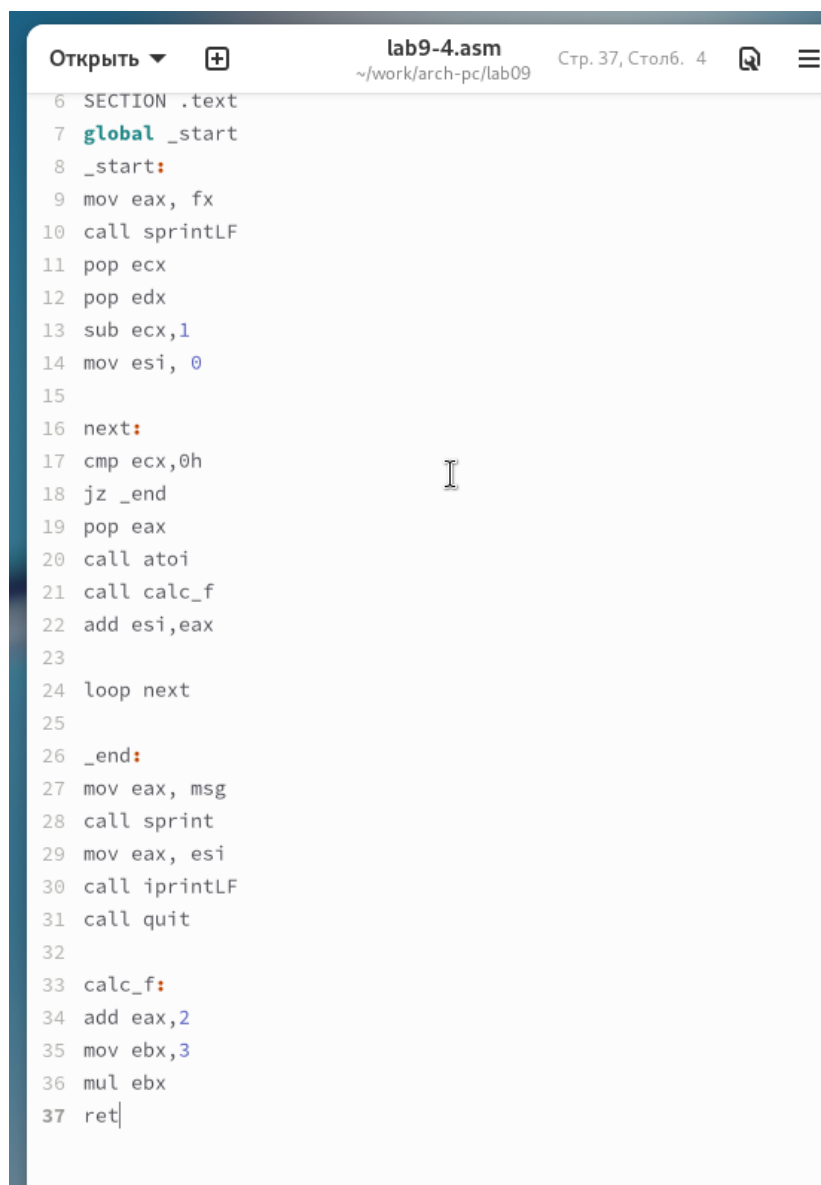
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd070: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd235: "/home/aidinakmuhammedov/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd267: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd270: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd272: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd27b: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd27d: "argument 3"
(gdb) c
Continuing.
argument
1
argument
2
argument 3
[Inferior 1 (process 4281) exited normally]
(gdb)
```

Рис. 2.15: Вывод значения регистра

Объясню, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12]) - шаг равен размеру переменной - 4 байтам.

- Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму.



```
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintf
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 call calc_f
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprintf
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 calc_f:
34 add eax,2
35 mov ebx,3
36 mul ebx
37 ret
```

Рис. 2.16: Программа в файле lab9-4.asm


```

aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-4.o -o lab9-4
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4
f(x)= 3(x + 2)
Результат: 0
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4 2
f(x)= 3(x + 2)
Результат: 12
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4 5
f(x)= 3(x + 2)
Результат: 21
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4 6 1 3 7 9
f(x)= 3(x + 2)
Результат: 108
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09$

```

Рис. 2.17: Запуск программы lab9-4.asm

7. В листинге приведена программа вычисления выражения  $(3 + 2) * 4 + 5$ . При запуске данная программа дает неверный результат. Проверил это. С помощью отладчика GDB, анализируя изменения значений регистров, определяю ошибку и исправлю ее.

Открыть ▾ 

lab9-5.asm  
~/work/arch-pc/lab09

Стр. 11, Столб. 10

```
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. 2.18: Код с ошибкой

```
aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-5

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 <_start+24>

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
0x80490f4 <_start+12>   mov     ecx,0x4
0x80490f9 <_start+17>   mul     ecx
0x80490fb <_start+19>   add     ebx,0x5
0x80490fe <_start+22>   mov     edi,ebx
>0x8049100 <_start+24> mov     eax,0x804a000
0x8049105 <_start+29>   call   0x804900f <sprint>
0x804910a <_start+34>   mov     eax,edi



native process 4382 (asm) In: _start L16 PC: 0x8049100
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции `add` и что по окончании работы в `edi` отправляется `ebx` вместо `eax`



Открыть ▾  lab9-5.asm Стр. 20, Столб. 10 

```
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. 2.20: Код исправлен

The screenshot shows a GDB terminal window with the title bar "aidinakmuhammedov@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-5". The window is divided into several sections. At the top, a register window shows "eax" with the value "25". Below it, a memory window shows addresses "ffffd0a0" and "x80490fe" with values "xffffd0a0" and "x80490fe <\_start+22>". The main window displays assembly code for the function "\_start":

```
0x80490fe <_start+22>  mov     edi,eax
0x8049100 <_start+24>  add     eax,eb804a000
0x8049105 <_start+29>  call    0x804900f <sprint>
0x804910a <_start+34>  mul     eax,edi
0x804910c <_start+36>  call    0x8049086 <iprintLF>
>0x8049111 <_start+41>  call    0x80490db <quit>
                                04a000
                                rint>
```

Below the assembly code, the GDB prompt shows the following commands and output:

```
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 4426) exited normally]
(gdb)
```

At the bottom right, a status bar shows "L14 PC: 0x80490fe" and "L?? PC: ??".

Рис. 2.21: Проверка работы

## **3 Выводы**

Освоили работу с подпрограммами и отладчиком.