

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: Игнатенкова Виктория Станиславовна

Группа: НММбд-02-24

МОСКВА

2024 г.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ.....	3
2. ЗАДАНИЕ.....	4
3. ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ.....	5
4. ВЫВОД.....	6

1. ЦЕЛЬ РАБОТЫ

Освоить навыки программирования с использованием подпрограмм и изучить базовые методы отладки с помощью GDB.

2. ЗАДАНИЕ

1. Понятие об отладке:

Опишите, что такое отладка и зачем она нужна.

2. Методы отладки:

Перечислите основные методы отладки программ.

3. Основные возможности отладчика GDB:

Опишите основные команды GD

В (с примерами).

4. Пошаговая отладка:

Проведите пошаговую отладку предоставленного кода с помощью GDB.

5. Реализация подпрограмм в NASM:

Напишите программу с использованием подпрограмм на языке ассемблера NASM.

6. Отладка программы с помощью GDB:

Отладьте написанную подпрограмму с помощью GDB.

3. ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

Создаём каталог для выполнения лабораторной работы № 9, перейдём в него и создаём файл lab09-1.asm:

```
vsignatenkova@dk8n63 ~ $ mkdir ~/work/arch-pc/lab09
vsignatenkova@dk8n63 ~ $ cd ~/work/arch-pc/lab09
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ touch lab09-1.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ls
lab09-1.asm
```

Рис.1. Каталог

Открываем файл и вносим листинг, затем запускаем:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gedit lab09-1.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -l lab09-1.lst lab09-1.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 3
2x+7=13
```

Рис.2. Проверка листинга

Вносим изменения и проводим проверку:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gedit lab09-1.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -l lab09-1.lst lab09-1.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 3
f(g(x))= 23
```

Рис.3. Проверка нового листинга

Создаём файл lab09-2.asm, вводим листинг и запускаем:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gedit lab09-2.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-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/>
```

Рис.4. Новый файл и запуск

Для более подробного анализа программы установим брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустим её и посмотрим дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start`:

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/v/s/vsignatenkova/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 5769) exited normally]
(gdb) break _start
Undefined command: "break _start". Try "help".
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/v/s/vsignatenkova/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.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
```

Рис.5. Анализ

Переключимся на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`:

```
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.6. Переключаемся

Перечислим различия отображения синтаксиса машинных команд в режимах АТТ и Intel. Включим режим псевдографики для более удобного анализа программы:

```
B+> 0x08049000 <_start>      mov     eax,0x4
    0x08049005 <_start+5>     mov     ebx,0x1
    0x0804900a <_start+10>    mov     ecx,0x804a000
    0x0804900f <_start+15>    mov     edx,0x8
    0x08049014 <_start+20>    int     0x80
    0x08049016 <_start+22>    mov     eax,0x4
    0x0804901b <_start+27>    mov     ebx,0x1
    0x08049020 <_start+32>    mov     ecx,0x804a008
    0x08049025 <_start+37>    mov     edx,0x7
    0x0804902a <_start+42>    int     0x80
    0x0804902c <_start+44>    mov     eax,0x1
    0x08049031 <_start+49>    mov     ebx,0x0
    0x08049036 <_start+54>    int     0x80

native process 5813 In: _start
(gdb) layout regs
(gdb)
```

Рис.7. Режим

Определим адрес предпоследней инструкции (mov ebx,0x0) и установим точку останова:

```
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000   lab09-2.asm:9
          breakpoint already hit 1 time
(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) █
```

Рис.8. Точка

Используем команду stepi (или si) для пошагового выполнения программы инструкция за инструкцией. После каждой команды stepi введём команду info registers (или сокращенно i r) для просмотра текущих значений регистров:

```
Num      Type           Disp Enb Address      What
eax              0x1              1
ecx              0x804a0081: file lab134520840 line 20.
edx              0x7              7
ebx      ype      0x1      Disp Enb Address      What
esp              0xfffffc370      0xfffffc370
ebp              0x0              0x0
esi              0x0              0
edi              0x0              0
eip              0x8049031      0x8049031 <_start+49>
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--
```

Рис.9. Команда si

Посмотрите значение переменной msg1 по имени

```
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
```

Рис.10. Переменная msg1

Изменим первый символ переменной msg1:

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

Рис.11. Msg1

С помощью команды set изменим значение регистра ebx:

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

Рис.12. Изменения

Чтобы посмотреть значения регистров используется команда print /F <val> (перед именем регистра обязательно ставится префикс \$):

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$3 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$4 = 2
```

Рис.13. Регистры

Завершаем выполнение программы с помощью команды continue (сокращенно c) или stepi (сокращенно si) и выйдем из GDB с помощью команды quit (сокращенно q):

```
End of assembler dump.
(gdb) layout asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $
```

Рис.14. Выход

Создаём исполняемый файл:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ls
in_out.asm lab09-1 lab09-1.asm lab09-1.lst lab09-1.o lab09-2 lab09-2.asm lab09-2.lst lab09-2.o lab09-3.asm
```

Рис.15. Файл

Для загрузки в gdb программы с аргументами необходимо использовать ключ `--args`.
Загрузим исполняемый файл в отладчик, указав аргументы:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-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"...
```

Рис.16. Использование ключа

Для начала установим точку останова перед первой инструкцией в программе и запустим ее:

```
Reading symbols from lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/v/s/vsignatenkova/
Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в 'ecx' количество
```

Рис.17. Точка

Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы):

```
8      pop ecx ; Извлекаем из
(gdb) x/x $esp
0xffffc330: 0x00000005
```

Рис.18. Адрес

```

(gdb) x/x $esp
0xfffffc330:      0x00000005
(gdb) x/s *(void**)( $esp + 4)
0xfffffc5a0:      "/afs/.dk.sci.pfu.edu.ru/home/v/s/vsignatena
(gdb) x/s *(void**)( $esp + 8)
0xfffffc5ea:      "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xfffffc5fc:      "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xfffffc60d:      "2"
(gdb) x/s *(void**)( $esp + 20)
0xfffffc60f:      "аргумент 3"
(gdb) x/s *(void**)( $esp + 24)
0x0:      <error: Cannot access memory at address 0x0>

```

Рис.19. Позиции стека

Первый адрес указывает на начало стека. Данные хранятся с шагом в 4 байта, поскольку размер стека ограничен 4 байтами. Для предотвращения конфликтов, каждый новый набор данных размещается в новом стеке.

САМОСТОЯТЕЛЬНАЯ РАБОТА

Создадим новый файл, чтобы выполнить задание:

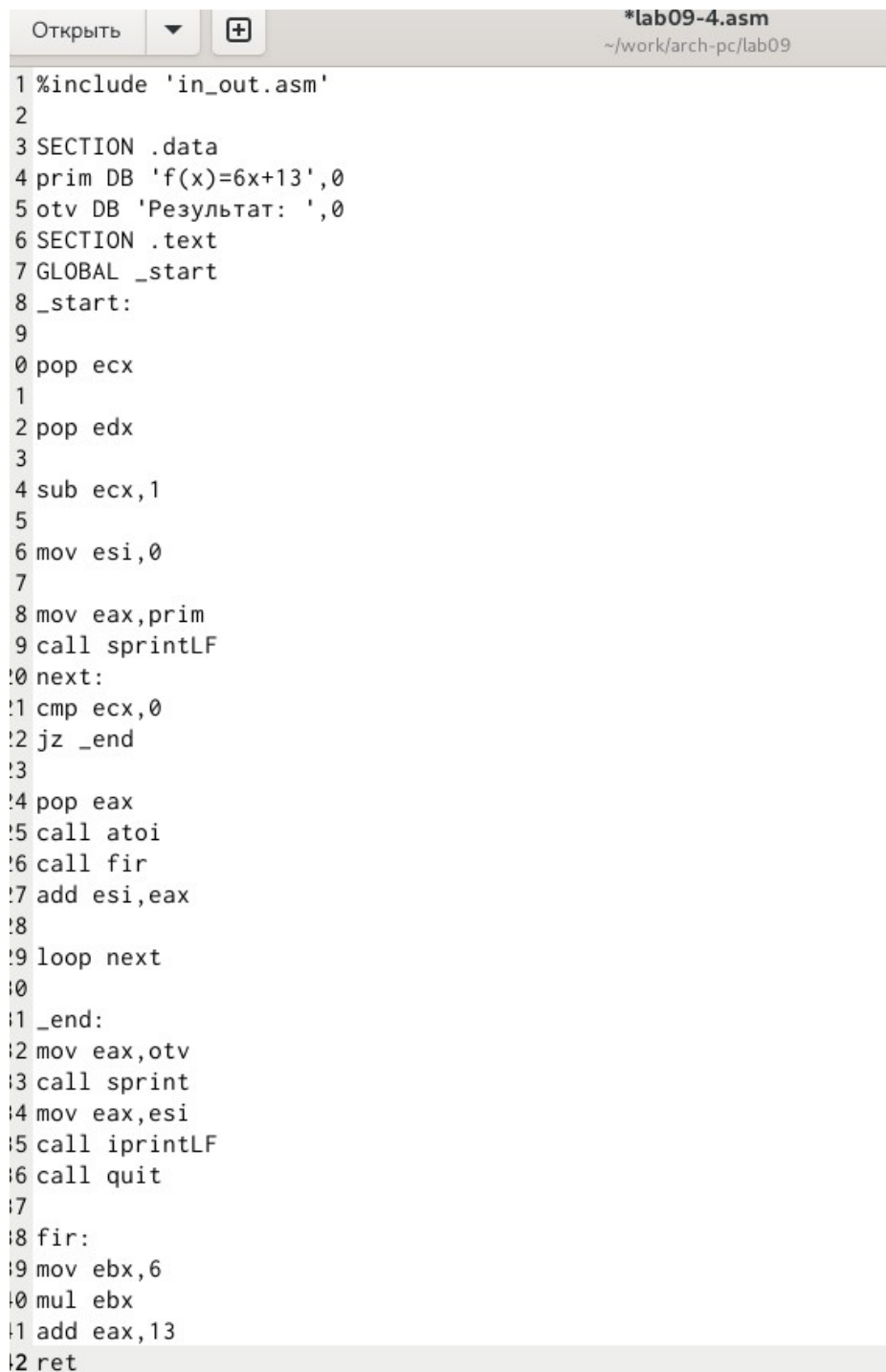
```

vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ touch lab09-4.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ды
bash: ды: команда не найдена
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1  lab09-1.asm  lab09-1.lst  lab09-1.o  lab09-2  lab09-2.asm  lab09-2.lst  lab09-2.o  lab09-3  lab09-3.asm  lab09-3.lst  lab09-3.o  lab09-4.asm

```

Рис.20. Файл

Преобразуем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму. У меня 15 вариантов, то есть $f(x)=13+6x$:



```
*lab09-4.asm
~/work/arch-pc/lab09

1 %include 'in_out.asm'
2
3 SECTION .data
4 prim DB 'f(x)=6x+13',0
5 otv DB 'Результат: ',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 pop ecx
11
12 pop edx
13
14 sub ecx,1
15
16 mov esi,0
17
18 mov eax,prim
19 call sprintf
20 next:
21 cmp ecx,0
22 jz _end
23
24 pop eax
25 call atoi
26 call fir
27 add esi,eax
28
29 loop next
30
31 _end:
32 mov eax,otv
33 call sprintf
34 mov eax,esi
35 call iprintf
36 call quit
37
38 fir:
39 mov ebx,6
40 mul ebx
41 add eax,13
42 ret
```

Рис.21. Программа

Проверим работу программы:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gedit lab09-4.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -l lab09-4.lst lab09-4.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-4 4 5 6 7
f(x)=6x+13
Результат: 184
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-4 10 11 12 13
f(x)=6x+13
Результат: 328
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $
```

Рис.22. Проверка

В листинге 9.3 приведена программа вычисления выражения $(3 + 2) * 4 + 5$. При запуске данная программа дает неверный результат. Проверим это:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ touch lab09-5.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gedit lab09-5.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -l lab09-5.lst lab09-5.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 10
```

Рис.23. Проверка. Ответ неверный, следовательно программа работает некорректно

С помощью отладчика GDB, анализируя изменения значений регистров, определим ошибку и исправим ее:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-5.lst lab09-5.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-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 lab09-5...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-5.asm, line 8.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/v/s/vsignatenkova/work/arch-pc/lab09/lab0
Breakpoint 1, _start () at lab09-5.asm:8
8      mov ebx,3
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:      mov     ebx,0x3
      0x080490ed <+5>:      mov     eax,0x2
      0x080490f2 <+10>:     add     ebx,eax
      0x080490f4 <+12>:     mov     ecx,0x4
      0x080490f9 <+17>:     mul     ecx
      0x080490fb <+19>:     add     ebx,0x5
      0x080490fe <+22>:     mov     edi,ebx
      0x08049100 <+24>:     mov     eax,0x804a000
      0x08049105 <+29>:     call   0x804900f <sprint>
      0x0804910a <+34>:     mov     eax,edi
      0x0804910c <+36>:     call   0x8049086 <iprintLF>
      0x08049111 <+41>:     call   0x80490db <quit>
End of assembler dump.
(gdb) █
```

Рис.24. Анализ

Изменяем регистры и повторно запускаем программу для проверки:

```
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf -l lab09-5.lst lab09-5.asm
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
vsignatenkova@dk8n63 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-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 lab09-5...
(No debugging symbols found in lab09-5)
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/v/s/vsignatenkova/work/arch-pc/lab09/lab09-5
Результат: 25
[Inferior 1 (process 15900) exited normally]
(gdb) █
```

Рис.25. Проверка исправленной программы

4. ВЫВОД

В результате изучения данного модуля освоили навыки программирования с использованием подпрограмм и изучили базовые методы отладки с помощью GDB.