

# **Отчет по лабораторной работе №9**

**Понятие подпрограммы. Отладчик GDB**

Краснопер Данила Олегович

# Содержание

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Цель работы</b>                    | <b>5</b>  |
| <b>2</b> | <b>Выполнение лабораторной работы</b> | <b>6</b>  |
| <b>3</b> | <b>Самостоятельная работа</b>         | <b>21</b> |
|          | <b>Вывод</b>                          | <b>27</b> |

## Список иллюстраций

|      |  |    |
|------|--|----|
| 2.1  | Создание каталога и файла . . . . .        | 6  |
| 2.2  | Текст программы . . . . .                  | 7  |
| 2.3  | Работа программы . . . . .                 | 8  |
| 2.4  | Измененный текст программы . . . . .       | 9  |
| 2.5  | Проверка работы программы . . . . .        | 10 |
| 2.6  | Текст второй программы . . . . .           | 11 |
| 2.7  | Отладка второго файла . . . . .            | 12 |
| 2.8  | Брежпоинт на метку _start . . . . .        | 12 |
| 2.9  | Дисассимплированный код . . . . .          | 13 |
| 2.10 | Intel'овское отображение . . . . .         | 14 |
| 2.11 | Псевдографика . . . . .                    | 14 |
| 2.12 | Наличие меток . . . . .                    | 15 |
| 2.13 | Просмотр регистров . . . . .               | 16 |
| 2.14 | Измененные регистры . . . . .              | 16 |
| 2.15 | Просмотр значения переменной . . . . .     | 17 |
| 2.16 | Значение переменной msg2 . . . . .         | 17 |
| 2.17 | Изменение значения переменной . . . . .    | 17 |
| 2.18 | Изменение msg2 . . . . .                   | 17 |
| 2.19 | Значение регистров ехх и еах . . . . .     | 18 |
| 2.20 | Значение регистров ебх . . . . .           | 18 |
| 2.21 | Завершение работы с файлов . . . . .       | 19 |
| 2.22 | Запуск файла в отладчике . . . . .         | 19 |
| 2.23 | Запуск файла lab10-3 через метку . . . . . | 19 |
| 2.24 | Адрес вершины стека . . . . .              | 20 |
| 2.25 | Все позиции стека . . . . .                | 20 |
| 3.1  | Текст программы . . . . .                  | 22 |
| 3.2  | Запуск программы . . . . .                 | 23 |
| 3.3  | Текст программы . . . . .                  | 24 |
| 3.4  | Запуск программы . . . . .                 | 24 |
| 3.5  | Запуск программы в отладчике . . . . .     | 25 |
| 3.6  | Анализ регистров . . . . .                 | 26 |
| 3.7  | Повторный запуск программы . . . . .       | 26 |

## Список таблиц

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

1) Я создал каталог lab9 и создал файл lab9-1.asm

```
dokrasnoper@dk3n62 ~ $ mkdir ~/work/arch-pc/lab09
dokrasnoper@dk3n62 ~ $ cd ~/work/arch-pc/lab09
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ touch lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ls
lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $
```

Рис. 2.1: Создание каталога и файла

2) Я ввел текст листинга в файл и запустил программу.

```

%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 ; 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

```

Рис. 2.2: Текст программы

```
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 5
2x+7=17
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ █
```

Рис. 2.3: Работа программы

3) Я изменил текст программы, чтобы она решала выражение  $f(g(x))$ .



```

lab9-1.asm      [----]  0 L:
SECTION .text
GLOBAL _start
_start:

mov  eax,prim1
call sprintLF

mov  eax,prim2
call sprintLF

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:

call _subcalcul

mov  ebx,2

```

Рис. 2.4: Измененный текст программы

```
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ./lab9-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 1
f(g(x))= 11
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ █
```

Рис. 2.5: Проверка работы программы

4) Я создал файл lab9-2.asm и вписал туда программу.

```
lab9-2.asm [-----] 8 L:
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
```

Рис. 2.6: Текст второй программы

5) Я загрузил и запустил файл второй программы в отладчик gdb.

```

dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ touch lab9-2.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ mc

dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.a
m
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ d -m elf_i386 -o lab9-2 lab9-2.o
bash: d: команда не найдена
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ gdb lab9-2
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/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 4978) exited normally]
(gdb)

```

Рис. 2.7: Отладка второго файла

6) Я поставил брекпоинт на метку `_start` и запустил программу.

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)

```

Рис. 2.8: Брекпоинт на метку `_start`

7) Я просмотрел дисассимплированный код программы начиная с метки.

```

(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. 2.9: Дисассимпированный код

- 8) С помощью команды я переключился на intel'овское отображение синтаксиса. Отличие заключается в командах, в дисассимпированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.

```

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

```

Рис. 2.10: Intel'овское отображение

9) Для удобства я включил режим псевдографики.

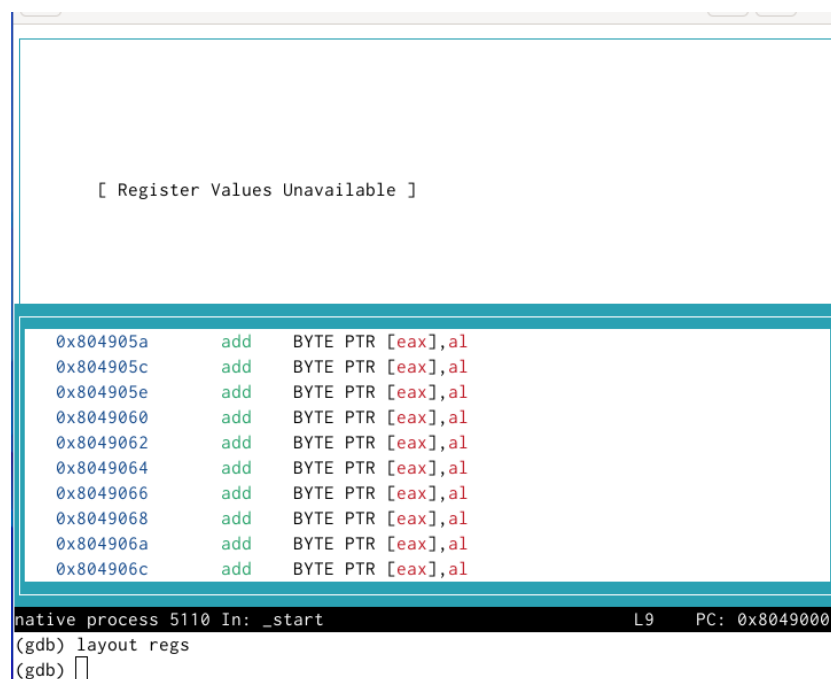


Рис. 2.11: Псевдографика

10) Я посмотрел наличие меток и добавил еще одну метку на предпоследнюю инструкцию.

```
[ Register Values Unavailable ]

0x8049dbc    add    BYTE PTR [eax],al
0x8049dbe    add    BYTE PTR [eax],al
0x8049dc0    add    BYTE PTR [eax],al
0x8049dc2    add    BYTE PTR [eax],al
0x8049dc4    add    BYTE PTR [eax],al
0x8049dc6    add    BYTE PTR [eax],al
0x8049dc8    add    BYTE PTR [eax],al
0x8049dca    add    BYTE PTR [eax],al
0x8049dcc    add    BYTE PTR [eax],al
0x8049dce    add    BYTE PTR [eax],al

native process 5110 In: _start          L9    PC: 0x8049000
(gdb) info breakpoints
Num    Type             Disp Enb Address      What
1      breakpoint       keep y  0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
(gdb) break *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. 2.12: Наличие меток

11) С помощью команды si я посмотрел регистры и изменил их.

```

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc4a0 0xffffc4a0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]

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 5381 In: _start L10 PC: 0x8049005
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab0
9/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
(gdb) si
(gdb) █

```

Рис. 2.13: Просмотр регистров

```

eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffc4a0 0xffffc4a0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049020 0x8049020 <_start+32>
eflags   0x202    [ IF ]

```

Рис. 2.14: Измененные регистры

12) С помощью команды я посмотрел значение переменной msg1.



```
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) █
```

Рис. 2.15: Просмотр значения переменной

13) Следом я посмотрел значение второй переменной msg2.

```
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "world!\n\034"
(gdb) █
```

Рис. 2.16: Значение переменной msg2

14) С помощью команды set я изменил значение переменной msg1.

```
(gdb) set {char}0x804a000='0'
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hh1lo, "
(gdb) █
```

Рис. 2.17: Изменение значения переменной

15) Я изменил переменную msg2.

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

Рис. 2.18: Изменение msg2

16) Я вывел значение регистров ехх и еах.

```
native process 14165 In: _start
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb)
```

Рис. 2.19: Значение регистров ecx и eax

- 17) Я изменил значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные.

```
native process 14165 In: _start
$5 = 0x0
(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb)
```

Рис. 2.20: Значение регистров ebx

18) Я завершил работу с файлов вышел.

```
0x08049031 <+49>:    mov     ebx, 0x0
0x08049036 <+54>:    int     0x80
End of assembler dump.
(gdb) layout regs
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $
```

Рис. 2.21: Завершение работы с файлов

19) Я скопировал файл lab8-2.asm и переименовал его. Запустил файл в отладчике и указал аргументы.

```
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ gdb --args lab9-3 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"...
Reading symbols from lab9-3...
(gdb)
```

Рис. 2.22: Запуск файла в отладчике

20) Поставил метку на \_start и запустил файл.

```
(gdb) b _start
Breakpoint 1 at 0x8049000: file lab9-3.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab09/lab9-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:9
9      mov     eax, 4
(gdb)
```

Рис. 2.23: Запуск файла lab10-3 через метку

21) Я проверил адрес вершины стека и убедился что там хранится 5 элементов.

```
(gdb) x/x $esp
0xfffffc430:      0x000000005
(gdb) █
```

---

Рис. 2.24: Адрес вершины стека

22) Я посмотрел все позиции стека. По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации.

```
(gdb) x/x $esp
0xfffffc430:      0x000000005
(gdb) x/s *(void**)(esp + 4)
0xfffffc681:      "/afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc6c8:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc6da:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xfffffc6eb:      "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc6ed:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
(gdb) █
```

---

Рис. 2.25: Все позиции стека

### **3 Самостоятельная работа**

- 1) Я преобразовал программу из лабораторной работы №8 и реализовал вычисления как подпрограмму.

```
sub ecx,1.

mov esi,0

mov eax,prim
call sprintLF
next:
cmp ecx, 0
jz _end.

pop eax
call atoi
call fir
add esi,eax

loop next.

_end:
mov eax,msg
call sprint
mov eax,esi
call iprintLF
call quit

fir:
mov ebx,10
mul ebx
sub eax,4
ret
```

Рис. 3.1: Текст программы

```
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ./lab9-4 2 3 5
f(x)=15x-9
Результат: 88
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ./lab9-4 1 1 1
f(x)=15x-9
Результат: 18
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $
```

Рис. 3.2: Запуск программы

- 2) Я переписал программу и попробовал запустить ее чтобы увидеть ошибку.  
Ошибка была арифметическая, так как вместо 25, программа выводит 10.

```

%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

```

Рис. 3.3: Текст программы

```

dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 10
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $

```

Рис. 3.4: Запуск программы

После появления ошибки, я запустил программу в отладчике.



```

dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ gdb lab9-5
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 lab9-5...
(No debugging symbols found in lab9-5)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab09/lab9-5

Breakpoint 1, 0x80490e8 in _start ()
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:    mov     $0x3,%ebx
                   ....:    #0x3 0x00000003

```

Рис. 3.5: Запуск программы в отладчике

Я открыл регистры и проанализировал их, понял что некоторые регистры стоят не на своих местах и исправил это.

```

eax      0x0      0
ebx      0x0      0
esp      0xffffc470 0xffffc470
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>

0x80490e0 <quit+5>    mov     eax,0x1
0x80490e5 <quit+10>   int     0x80
0x80490e7 <quit+12>   ret
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

native process 18608 In: _start L?? PC: 0x80490e8
(gdb) layout regs
(gdb) layout asm
(gdb) layout regs
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab0
9/lab9-5

Breakpoint 1, 0x080490e8 in _start ()

```

Рис. 3.6: Анализ регистров

Я изменил регистры и запустил программу, программа вывела ответ 25, то есть все работает правильно.

```

<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-5...
(No debugging symbols found in lab9-5)
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab0
9/lab9-5
Результат: 25
[Inferior 1 (process 18843) exited normally]
(gdb) █

```

Рис. 3.7: Повторный запуск программы

## Вывод

Я приобрел навыки написания программ использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями.