

# **Лабораторная работа №10**

**Понятие подпрограммы. Отладчик GDB.**

Воробчук Лилия Андреевна

# 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>18</b> |
| <b>4</b> | <b>Выводы</b>                         | <b>24</b> |

# Список иллюстраций

|      |  |    |
|------|--|----|
| 2.1  | создание файла . . . . .                   | 6  |
| 2.2  | листинг . . . . .                          | 7  |
| 2.3  | создание файла . . . . .                   | 8  |
| 2.4  | внесение изменений . . . . .               | 9  |
| 2.5  | запуск . . . . .                           | 10 |
| 2.6  | ввод листинга . . . . .                    | 10 |
| 2.7  | отладчик . . . . .                         | 11 |
| 2.8  | брекпоинт . . . . .                        | 11 |
| 2.9  | дисассимплированный код . . . . .          | 12 |
| 2.10 | Intel'овское отображение . . . . .         | 12 |
| 2.11 | режим псевдографики . . . . .              | 13 |
| 2.12 | Наличие меток . . . . .                    | 13 |
| 2.13 | просмотр регистров . . . . .               | 13 |
| 2.14 | Просмотр значения переменной . . . . .     | 13 |
| 2.15 | Изменение значения переменной . . . . .    | 14 |
| 2.16 | Изменение msg2 . . . . .                   | 14 |
| 2.17 | Значение регистров ехх и еах . . . . .     | 15 |
| 2.18 | Значение регистров ебх . . . . .           | 16 |
| 2.19 | Запуск файла в отладчике . . . . .         | 16 |
| 2.20 | Запуск файла lab10-3 через метку . . . . . | 17 |
| 2.21 | Адрес вершины стека . . . . .              | 17 |
| 2.22 | Все позиции стека . . . . .                | 17 |
| 3.1  | Текст программы . . . . .                  | 19 |
| 3.2  | Запуск программы . . . . .                 | 20 |
| 3.3  | Текст програмы . . . . .                   | 21 |
| 3.4  | Запуск программы . . . . .                 | 21 |
| 3.5  | Запуск программы в отладчике . . . . .     | 22 |
| 3.6  | Анализ регистров . . . . .                 | 23 |
| 3.7  | Повторный запуск программы . . . . .       | 23 |

## 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. Создаю каталог lab10 и файл lab10-1.asm (рис. 2.1)

```
lavorobchuk@dk8n64 ~ $ mkdir ~/work/arch-pc/lab10  
lavorobchuk@dk8n64 ~ $ cd ~/work/arch-pc/lab10  
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ touch lab10-1.asm
```

Рис. 2.1: создание файла

2. Ввожу текст листинга (рис. 2.2)

```

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

```

Рис. 2.2: листинг

### 3. Запускаю программу (рис. 2.3)

```
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf lab10-1.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ./lab10-1
Введите x: 4
2x+7=15
```

Рис. 2.3: создание файла

4. Меняю текст программы, чтобы она решала выражение  $f(g(x))$ . (рис. 2.4)



```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 prim1: DB 'f(x)=2x+7',0
5 prim2: DB 'g(x)=3x-1',0
6 result: DB 'f(g(x))=',0
7 SECTION .bss
8 x: RESB 80
9 rez: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13
14 mov eax, prim1
15 call sprintLF
16
17 mov eax, prim2
18 call sprintLF
19
20 mov eax, msg
21 call sprintLF
22
23 mov ecx, x
24 mov edx, 80
25 call sread
26 mov eax,x
27 call atoi
28 call _calcul
29 mov eax,result
30 call sprint
31 mov eax,[rez]
32 call iprintLF
33 call quit
34
35 _calcul:
36
37 call _subcalcul
38 mov ebx,2
39 mul ebx
40 add eax,7
41 mov [rez],eax
42 ret
43
44 _subcalcul:
45 mov ebx, 3
46 mul ebx
47 sub eax, 1
48 ret

```

Рис. 2.4: внесение изменений

5. Запускаю измененную программу (рис. 2.5)

```
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf lab10-1.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ./lab10-1
f(x)=2x+7
g(x)=3x-1
Введите x:
4
f(g(x))=29
```

Рис. 2.5: запуск

6. Создаю файл lab10-2.asm и вписываю туда Листинг 10.2(рис. 2.6)

```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1Len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2Len: equ $ - msg2
6 SECTION .text
7 global _start
8 _start:
9 mov eax, 4
10 mov ebx, 1
11 mov ecx, msg1
12 mov edx, msg1Len
13 int 0x80
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, msg2
17 mov edx, msg2Len
18 int 0x80
19 mov eax, 1
20 mov ebx, 0
21 int 0x80
```

Рис. 2.6: ввод листинга

7. Запускаю файл второй программы в отладчик gdb. (рис. 2.7)

```
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-2.lst lab10-2.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-2 lab10-2.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ gdb lab10-2
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 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 lab10-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/l/a/lavorobchuk/work/arch-pc/lab10/lab10-2
Hello, world!
[Inferior 1 (process 1168726) exited normally]
(gdb)
```

Рис. 2.7: отладчик

8. Ставлю брекпоинт на метку \_start и запускаю программу(рис. 2.8)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/l/a/lavorobchuk/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:9
9      mov eax, 4
```

Рис. 2.8: брекпоинт

9. Просматриваю дисассимплированный код программы, начиная с метки.(рис. 2.9)

```
(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.
```

Рис. 2.9: дисассимплированный код

10. С помощью команды переключаюсь на intel'овское отображение синтаксиса. Отличие заключается в командах, в дисассимлированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. (рис. 2.10)

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _star
No symbol "_star" in current context.
(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. 2.10: Intel'овское отображение

11. Для удобства включаю режим псевдографики. (рис. 2.11)

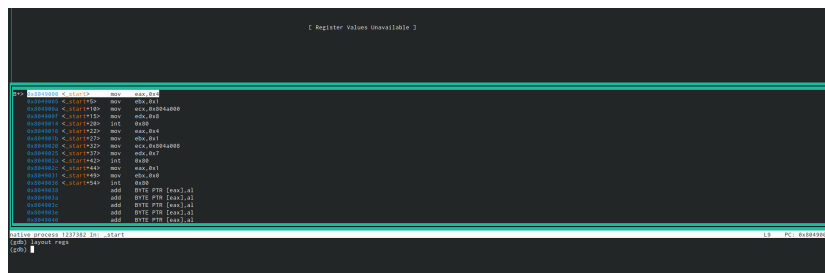


Рис. 2.11: режим псевдографики

12. Посматриваю наличие меток и добавляю еще одну метку на предпоследнюю инструкцию. (рис. 2.12)

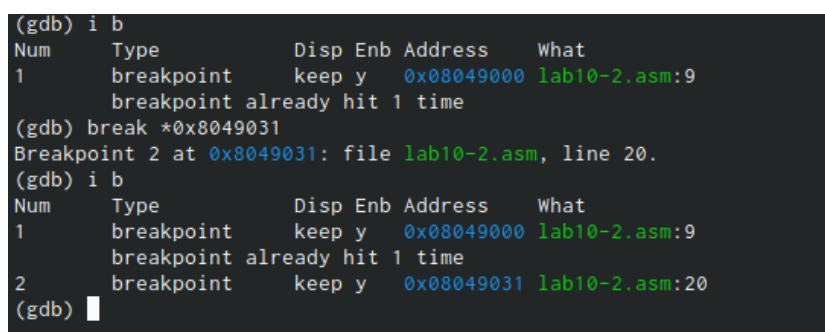


Рис. 2.12: Наличие меток

13. С помощью команды si посматриваю регистры и изменяю их (рис. 2.13)

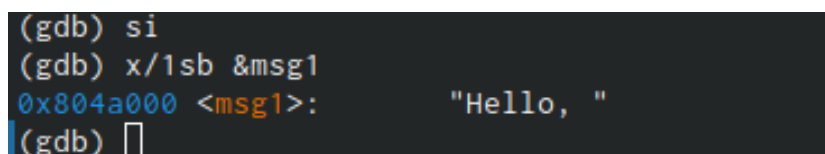


Рис. 2.13: просмотр регистров

14. Смотрю значение второй переменной msg2. (рис. 2.14)

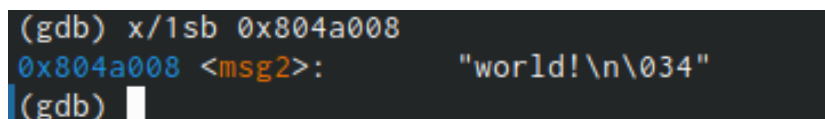


Рис. 2.14: Просмотр значения переменной

15. С помощью команды set меняю значение переменной msg1. (рис. 2.15)

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

Рис. 2.15: Изменение значения переменной

16. Меняю переменную msg2.(рис. 2.16)

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

Рис. 2.16: Изменение msg2

17. Вывожу значение регистров ехх и еах (рис. 2.17)

```
(gdb) p/f $msg1
$1 = void
(gdb) p/s $eax
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb)
```

Рис. 2.17: Значение регистров ecx и eax

18. Меняю 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. (рис. 2.18)

```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb)

```

Рис. 2.18: Значение регистров ebx

19. Копирую файл lab9-2.asm и переименовываю его. Запускаю файл в отладчике и указываю аргументы.(рис. 2.19)

```

lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ cp ~/work/arch-pc/lab09/lab9-2.asm ~/work/arch-pc/lab10/lab10-3.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-3.lst lab10-3.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-3 lab10-3.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ gdb --args lab10-3 2 7 '4'
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 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 lab10-3...
(gdb)

```

Рис. 2.19: Запуск файла в отладчике

20. Ставлю метку на \_start и запускаю файл.(рис. 2.20)



```

lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ cp ~/work/arch-pc/lab09/lab9-2.asm ~/work/arch-pc/lab10/lab10-3.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-3.lst lab10-3.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-3 lab10-3.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ gdb --args lab10-3 2 7 '4'
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 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 lab10-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/l/a/lavorobchuk/work/arch-pc/lab10/lab10-3 2 7 4

Breakpoint 1, _start () at lab10-3.asm:5
5      pop ecx
(gdb)

```

Рис. 2.20: Запуск файла lab10-3 через метку

21. Проверяю адрес вершины стека и убеждаюсь, что там хранится 5 элементов.(рис. 2.21)

```

(gdb) x/x $esp
0xfffffc660:      0x00000004
(gdb)

```

Рис. 2.21: Адрес вершины стека

22. Смотрю 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. (рис. 2.22)

```

0xfffffc660:      0x00000004
(gdb) x/s *(void**)(esp + 4)
0xfffffc8d1:      "/afs/.dk.sci.pfu.edu.ru/home/l/a/lavorobchuk/work/arch-pc/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc919:      "2"
(gdb) x/s *(void**)(esp + 12)
0xffffc91b:      "7"
(gdb) x/s *(void**)(esp + 16)
0xffffc91d:      "4"
(gdb) x/s *(void**)(esp + 20)
0x0:      <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)(esp + 24)
0xffffc91f:      "SHELL=/bin/bash"
(gdb)

```

Рис. 2.22: Все позиции стека

### **3 Самостоятельная работа**

1. Преобразовываю программу из лабораторной работы №9 и реализовываю вычисления как подпрограмму.(рис. 3.1)

```

1 %include 'in_out.asm'
2
3 SECTION data
4 primer DB 'f (x) = 4x+3',0
5 result 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,primer
19 call sprintLF
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,result
33 call sprint
34 mov eax,esi
35 call iprintLF
36 call quit
37
38 fir:
39 mov ebx, 4
40 mul ebx
41 add eax, 3
42 ret
43

```

Рис. 3.1: Текст программы

2. Создаю исполняемый файл и проверяю его работу. (рис. 3.2)

```
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-4.lst lab10-4.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-4 lab10-4.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ./lab10-4 2 5 3 7 11
f(x) = 4x+3
Результат: 127
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $
```

Рис. 3.2: Запуск программы

3. Ввожу в файл Листинг 10.3 (рис. 3.3)

```

1 %include 'in_out.asm'
2
3 SECTION data
4 div: DB 'Результат: ',0
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov ebx,3
11 mov eax, 2
12 add ebx,eax
13 mov ecx,4
14 mul ecx
15 add ebx,5
16 mov edi, ebx
17
18 mov eax,div
19 call sprint
20 mov eax, edi
21 call iprintLF
22
23 call quit
24

```

Рис. 3.3: Текст программы

4. Запускаю программу и вижу, что результат вычисления неверный(рис. 3.4)

```

lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ touch backstreet.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf backstreet.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o backstreet backstreet.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ./backstreet
Результат: 10

```

Рис. 3.4: Запуск программы

5. После появления ошибки запускаю программу в отладчике (рис. 3.5)

```
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o backstreet backstreet.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ gdb backstreet
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 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 backstreet...
(No debugging symbols found in backstreet)
(gdb) b _start
Breakpoint 1 at 0x080490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/1/a/lavorobchuk/work/arch-pc/lab10/backstreet

Breakpoint 1, 0x080490e8 in _start ()
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
No symbol table is loaded. Use the "file" command.
(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    0x0804900f <sprint>
    0x0804910a <+34>: mov     eax,edi
    0x0804910c <+36>: call    0x08049086 <iprintLF>
    0x08049111 <+41>: call    0x080490db <quit>
End of assembler dump.
(gdb)
```

Рис. 3.5: Запуск программы в отладчике

6. Открываю регистры и, проанализировав их, понимаю, что некоторые регистры стоят не на своих местах и исправляю это (рис. 3.6)

```
0x80490d6 <atoi.restore>    pop     esi
0x80490d7 <atoi.restore+1>    pop     edx
0x80490d8 <atoi.restore+2>    pop     ecx
0x80490d9 <atoi.restore+3>    pop     ebx
0x80490da <atoi.restore+4>    ret
0x80490db <quit>              mov     ebx,0x0
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
0x80490fe <_start+22>       mov     edi,ebx
0x8049100 <_start+24>       mov     eax,0x804a000

native process 2004107 In: _start
(gdb) layout regs
(gdb) █
```

Рис. 3.6: Анализ регистров

7. Изменяю регистры и запускаю программу, программа вывела ответ 25, то есть все работает правильно. (рис. 3.7)

```
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ nasm -f elf backstreet.asm
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o backstreet backstreet.o
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ ./backstreet
Результат: 25
lavorobchuk@dk8n64 ~/work/arch-pc/lab10 $ █
```

Рис. 3.7: Повторный запуск программы

## 4 Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм, ознакомилась с методами отладки при помощи GDB и его основными возможностями.