

Отчет по лабораторной работе №10

Понятие подпрограммы. Отладчик GDB

Лушин Артем Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работа	20
	Вывод	26

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	13
2.11	Псевдографика	14
2.12	Наличие меток	14
2.13	Просмотр регистров	15
2.14	Измененные регистры	15
2.15	Просмотр значения переменной	16
2.16	Значение переменной msg2	16
2.17	Изменение значения переменной	16
2.18	Изменение msg2	16
2.19	Значение регистров ехх и еах	17
2.20	Значение регистров ебх	17
2.21	Завершение работы с файлов	17
2.22	Запуск файла в отладчике	18
2.23	Запуск файла lab10-3 через метку	18
2.24	Адрес вершины стека	18
2.25	Все позиции стека	19
3.1	Текст программы	21
3.2	Запуск программы	22
3.3	Текст программы	23
3.4	Запуск программы	24
3.5	Запуск программы в отладчике	24
3.6	Анализ регистров	25
3.7	Повторный запуск программы	25

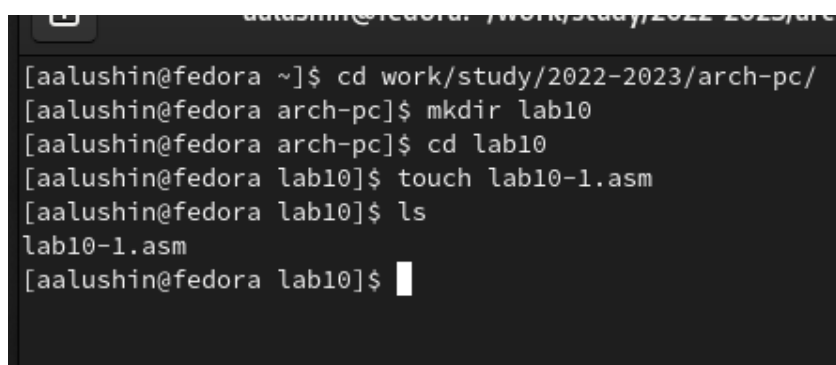
Список таблиц

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 lab10 и создал файл lab10-1.asm

A screenshot of a terminal window with a dark background. The terminal shows a series of commands and their outputs. The user is 'aalushin' on a 'fedora' system. The commands are: 'cd work/study/2022-2023/arch-pc/' to change the directory, 'mkdir lab10' to create a new directory, 'cd lab10' to enter the new directory, 'touch lab10-1.asm' to create a new file, and 'ls' to list the contents. The output of 'ls' shows 'lab10-1.asm'. The prompt returns to '[aalushin@fedora lab10]\$' after the last command.

```
[aalushin@fedora ~]$ cd work/study/2022-2023/arch-pc/  
[aalushin@fedora arch-pc]$ mkdir lab10  
[aalushin@fedora arch-pc]$ cd lab10  
[aalushin@fedora lab10]$ touch lab10-1.asm  
[aalushin@fedora lab10]$ ls  
lab10-1.asm  
[aalushin@fedora lab10]$
```

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

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

```

Рис. 2.2: Текст программы

```
[a]# Остановка /usr/bin/mc -f $MC_TMP_FILE -f@  
[aalushin@fedora lab10]$ nasm -f elf lab10-1.asm  
[aalushin@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o  
[aalushin@fedora lab10]$ ./lab10-1  
Введите x: 5  
2x+7=17
```

Рис. 2.3: Работа программы

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


```

#include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
prim1: DB 'f(x) = 2x+7',0
prim2: DB 'g(x) = 3x-1',0
result: DB 'f(g(x))= ',0

SECTION .bss
x: RESB 80
res: RESB 80

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

```

Рис. 2.4: Измененный текст программы

```
[aalushin@fedora lab10]$ nasm -f elf lab10-1.asm
[aalushin@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[aalushin@fedora lab10]$ ./lab10-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 1
f(g(x))= 11
[aalushin@fedora lab10]$
```

Рис. 2.5: Проверка работы программы

4) Я создал файл lab10-2.asm и вписал туда программу.

```
lab10-2.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
```

Рис. 2.6: Текст второй программы

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

```
[aalushin@fedora lab10]$ nasm -f elf -g -l lab10-2.lst lab10-2.asm
[aalushin@fedora lab10]$ ld -m elf_i386 -o lab10-2 lab10-2.o
[aalushin@fedora lab10]$ gdb lab10-2
GNU gdb (GDB) Fedora 12.1-1.fc36
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-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 lab10-2...
(gdb) r
Starting program: /home/aalushin/work/study/2022-2023/arch-pc/lab10/lab10-2

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Hello, world!
[Inferior 1 (process 3781) 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 lab10-2.asm, line 11.
(gdb) r
Starting program: /home/aalushin/work/study/2022-2023/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:11
11      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.
```

Рис. 2.9: Дисассимплированный код

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

```
(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.
```

Рис. 2.10: Intel'овское отображение

- 9) Для удобства я включил режим псевдографики.

```

[ Register Values Unavailable ]

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
0x804902c <_start+44>  mov    eax,0x1
0x8049031 <_start+49>  mov    ebx,0x0
0x8049036 <_start+54>  int     0x80

: No process In: L7
b) layout regs
b)

```

Рис. 2.11: Псевдографика

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

```

0x8049025 <_start+37>  mov    ecx,0x1
0x804902a <_start+42>  int     0x80
0x804902c <_start+44>  mov    eax,0x1
b+ 0x8049031 <_start+49>  mov    ebx,0x0
0x8049036 <_start+54>  int     0x80
0x8049038             add    BYTE PTR [eax],al

native process 3918 In: _start L11 PC: 0x8049038
(gdb) layout regs
(gdb) info breakpoints
Num    Type             Disp Enb Address      What
1      breakpoint       keep y  0x08049000 lab10-2.asm:11
      breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab10-2.asm, line 26.
(gdb) i b
Num    Type             Disp Enb Address      What
1      breakpoint       keep y  0x08049000 lab10-2.asm:11
      breakpoint already hit 1 time
2      breakpoint       keep y  0x08049031 lab10-2.asm:26
(gdb)

```

Рис. 2.12: Наличие меток

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

```

eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

B+ 0x8049000 <_start> mov $0x4,%eax
> 0x8049005 <_start+5> mov $0x1,%ebx
0x804900a <_start+10> mov $0x804a000,%ecx
0x804900f <_start+15> mov $0x8,%edx
0x8049014 <_start+20> int $0x80
0x8049016 <_start+22> mov $0x4,%eax
0x804901b <_start+27> mov $0x1,%ebx
0x8049020 <_start+32> mov $0x804a008,%ecx
0x8049025 <_start+37> mov $0x7,%edx
0x804902a <_start+42> int $0x80
0x804902c <_start+44> mov $0x1,%eax
0x8049031 <_start+49> mov $0x0,%ebx
0x8049036 <_start+54> int $0x80
0x8049038      add %al,(%eax)

native process 3985 In: _start L12 PC: 0x
(gdb) layout regs
(gdb) i r
The program has no registers now.
(gdb) run
Starting program: /home/aalushin/work/study/2022-2023/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:11
(gdb) si
(gdb)

```

Рис. 2.13: Просмотр регистров

```

eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43

```

Рис. 2.14: Измененные регистры

12) С помощью команды `y` 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 0x804a008
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}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hhllo, "
(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 еsx и еax.


```

(gdb) p/f $msg1
$2 = void
(gdb) p/s $eax
$3 = 4
(gdb) p/t $eax
$4 = 100
(gdb) p/c $ecx
$5 = 0 '\000'
(gdb) p/x $ecx
$6 = 0x0

```

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.

```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$7 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$8 = 2

```

Рис. 2.20: Значение регистров ebx

- 18) Я завершил работу с файлов вышел.

```

[Inferior 1 (process 3985) exited normally]

```

Рис. 2.21: Завершение работы с файлов

- 19) Я скопировал файл lab9-2.asm и переименовал его. Запустил файл в отладчике и указал аргументы.

```
[aalushin@fedora lab10]$ mc
[aalushin@fedora lab10]$ gdb --args lab10-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora 12.1-1.fc36
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-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 lab10-3...
(gdb)
```

Рис. 2.22: Запуск файла в отладчике

20) Поставил метку на `_start` и запустил файл.

```
Reading symbols from lab10-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 6.
(gdb) r
Starting program: /home/aalushin/work/study/2022-2023/arch-pc/lab10/lab10-3 аргумент1 аргумент 2
аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
^C Cancelling download of separate debug info for system-supplied DSO at 0xf7ffc000...

Breakpoint 1, _start () at lab10-3.asm:6
6      pop ecx
(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
0xffffd180: 0x00000005
(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/s *(void**)($esp + 4)
0xfffffd333:    "/home/aalushin/work/study/2022-2023/arch-pc/lab10/lab10-3"
(gdb) x/s *(void**)($esp + 8)
0xfffffd36d:    "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xfffffd37f:    "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xfffffd390:    "2"
(gdb) x/s *(void**)($esp + 20)
0xfffffd392:    "аргумент 3"
(gdb) x/s *(void**)($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 №9 и реализовал вычисления как подпрограмму.

```

#include 'in_out.asm'

SECTION .data
prim DB 'f(x)=2x+15',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

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,otv
call sprint
mov eax,esi
call iprintLF
call quit

fir:
mov ebx,2
mul ebx
add eax,15
ret

```

Рис. 3.1: Текст программы

```
[aalushin@fedora lab10]$ ./4 1 2 3
f(x)=2x+15
Результат: 57
[aalushin@fedora lab10]$ ./4 1 2 3 4
f(x)=2x+15
Результат: 80
[aalushin@fedora lab10]$
```

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.

```
5.asm [-M--
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

SECTION .text
GLOBAL _start
_start:

mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx

mov eax,div
call sprint
mov eax,edi
call iprintLF

call quit
```

Рис. 3.3: Текст программы

```
[aalushin@fedora lab10]$ nasm -f elf 5.asm
[aalushin@fedora lab10]$ ld -m elf_i386 -o 5 5.o
[aalushin@fedora lab10]$ ./5
Результат: 10
```

Рис. 3.4: Запуск программы

После появления ошибки, я запустил программу в отладчике.

```
[aalushin@fedora lab10]$ ld -m elf_i386 -o 5 5.o
[aalushin@fedora lab10]$ gdb 5
GNU gdb (GDB) Fedora 12.1-1.fc36
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-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 5...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file 5.asm, line 10.
(gdb) r
Starting program: /home/aalushin/work/study/2022-2023/arch-pc/lab10/5

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Download failed: Нет маршрута до узла. Continuing without debug info for system-supplied
xf7ffc000.

Breakpoint 1, _start () at 5.asm:10
10      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 <sprintf>
```

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.


```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
0x8049105 <_start+29> call    0x804900f <sprint>
0x804910a <_start+34> mov     eax,edi

```

Рис. 3.6: Анализ регистров

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

```

[aalushin@fedora lab10]$ nasm -f elf -g -l 5.lst 5.asm
[aalushin@fedora lab10]$ ld -m elf_i386 -o 5 5.o
[aalushin@fedora lab10]$ gdb 5
GNU gdb (GDB) Fedora 12.1-1.fc36
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-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 5...
(gdb) r
Starting program: /home/aalushin/work/study/2022-2023/arch-pc/lab10/5

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Download failed: Нет маршрута до узла. Continuing without debug info for system-supplied DSO at 0
xf7ffc000.
Результат: 25
[Inferior 1 (process 3929) 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.