

# **Отчёт по лабораторной работе №9**

**Дисциплина: Архитектура компьютера**

Ардеев Никита Евгеньевич НММбд-01-23

# 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>Задания для самостоятельной работы</b>	<b>13</b>
<b>4</b>	<b>Выводы</b>	<b>17</b>

# Список иллюстраций

2.1	- . . . . .	6
2.2	- . . . . .	6
2.3	- . . . . .	7
2.4	- . . . . .	7
2.5	- . . . . .	8
2.6	- . . . . .	8
2.7	- . . . . .	9
2.8	- . . . . .	9
2.9	- . . . . .	10
2.10	- . . . . .	10
2.11	- . . . . .	10
2.12	- . . . . .	11
2.13	- . . . . .	11
2.14	- . . . . .	12
3.1	- . . . . .	14
3.2	- . . . . .	15
3.3	- . . . . .	15
3.4	- . . . . .	16

## 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

Создал каталог для выполнения лабораторной работы No 9, перешел в него и создал файл lab09-1.asm, ввел в него текст программы из листинга 9.1. Создал исполняемый файл и проверил его работу: (рис. [2.1])

```
neardeev@dk4n60 ~ $ mkdir ~/work/arch-pc/lab09
neardeev@dk4n60 ~ $ cd ~/work/arch-pc/lab09
neardeev@dk4n60 ~/work/arch-pc/lab09 $ touch lab09-1.asm
neardeev@dk4n60 ~/work/arch-pc/lab09 $ mc\
>

neardeev@dk4n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
neardeev@dk4n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
neardeev@dk4n60 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 6
2x+7=19
neardeev@dk4n60 ~/work/arch-pc/lab09 $
```

Рис. 2.1: -

Изменил текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения  $\boxtimes(\boxtimes(\boxtimes))$  и проверил его работу(рис. [2.2])

```
neardeev@dk4n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
neardeev@dk4n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
neardeev@dk4n60 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 6
f(g(x))=41
neardeev@dk4n60 ~/work/arch-pc/lab09 $
```

Рис. 2.2: -

Создайте файл lab09-2.asm с текстом программы из Листинга 9.2, получил исполняемый файл.(рис. [-2.3])

```
f(g(x))=41
neardeev@dk4n60 ~/work/arch-pc/lab09 $ touch lab09-2.asm
neardeev@dk4n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-2.asm
neardeev@dk4n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
neardeev@dk4n60 ~/work/arch-pc/lab09 $ ./lab09-2
Hello, world!
neardeev@dk4n60 ~/work/arch-pc/lab09 $
```

```
19 mov eax, 1
20 mov ebx, 0
21 int 0x80
```

Рис. 2.3: -

Добавил отладочную информацию в исполняемый файл, загрузил исполняемый файл в отладчик gdb и проверил работу программы(рис. [2.4])

```
neardeev@dk4n60 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
neardeev@dk4n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
neardeev@dk4n60 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 12.1 vanilla) 12.1
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 lab09-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/n/e/neardeev/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 8801) exited normally]
(gdb)
```

Рис. 2.4: -

Установил брейкпоинт на метку \_start, посмотрел дисассимилированный код программы, переключился на отображение команд с Intel'овским синтаксисом. Отличие в том, что в АТТ первые аргументы всех команд записаны в виде 16-ричного числа, а в intel так записываются адреса вторых аргументов(рис. [2.5])

```

(gdb) break _start
Breakpoint 1 at 0x0049000: file lab09-2.asm, line 9.
(gdb) disassemble _start
Dump of assembler code for function _start:
0x0049000 <+0>: mov    $0x4,%eax
0x0049005 <+5>: mov    $0x1,%ebx
0x004900a <+10>: mov    $0x804a000,%ecx
0x004900f <+15>: mov    $0x8,%edx
0x0049014 <+20>: int    $0x80
0x0049016 <+22>: mov    $0x4,%eax
0x004901b <+27>: mov    $0x1,%ebx
0x0049020 <+32>: mov    $0x804a000,%ecx
0x0049025 <+37>: mov    $0x7,%edx
0x004902a <+42>: int    $0x80
0x004902c <+44>: mov    $0x1,%eax
0x0049031 <+49>: mov    $0x0,%ebx
0x0049036 <+54>: int    $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
0x0049000 <+0>: mov    eax,0x4
0x0049005 <+5>: mov    ebx,0x1
0x004900a <+10>: mov    ecx,0x804a000
0x004900f <+15>: mov    edx,0x8
0x0049014 <+20>: int    0x80
0x0049016 <+22>: mov    eax,0x4
0x004901b <+27>: mov    ebx,0x1
0x0049020 <+32>: mov    ecx,0x804a000
0x0049025 <+37>: mov    edx,0x7

```

Рис. 2.5: -

Включил режим псевдографики для более удобного анализа программы (рис. [2.6])

```

[ Register Values Unavailable ]

b* 0x0049000 <_start> mov    eax,0x4
0x0049005 <_start+5> mov    ebx,0x1
0x004900a <_start+10> mov    ecx,0x804a000
0x004900f <_start+15> mov    edx,0x8
0x0049014 <_start+20> int    0x80
0x0049016 <_start+22> mov    eax,0x4
0x004901b <_start+27> mov    ebx,0x1
0x0049020 <_start+32> mov    ecx,0x804a000
0x0049025 <_start+37> mov    edx,0x7

Exec No process in:
(gdb) layout regs
(gdb)

```

Рис. 2.6: -

Проверил точку останова с помощью команды info breakpoints,установил еще одну точку останова по адресу инструкции, проверил это(рис. [2.7])



```

[ Register Values Unavailable ]

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80

exec No process in:
(gdb) info breakpoints
Num    Type             Disp Enb Address      What
1      breakpoint      keep y  0x08049000 lab09-2.asm:9
(gdb) b *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
2      breakpoint      keep y  0x08049031 lab09-2.asm:20
(gdb)

```

Рис. 2.7: -

Выполнил 5 инструкций с помощью команды stepi, меняются значения регистров eax, ebx, edx, ecx, eip, проверил это с помощью команды i r(рис. [2.8])

```

File  Правка  Вид  Закладки  Модули  Настройка  Справка
[+] Новая вкладка  [x] Разделить окно
[?] Копировать  [V] Вставить  [F] Найти

Register group: general
eax    0x8      8      ecx    0x804a000  134520832
edx    0x8      8      ebx    0x1        1
esp    0xffffc330 0xffffc330  ebp    0x0        0
esi    0x0      0      edi    0x0        0
eip    0x8049016 0x8049016 <_start+22>  eflags  0x202      [ IF ]
cs     0x23     35     ss     0x2b     43
fs     0x20     43     gs     0x2b     43
fs     0x0      0      gs     0x0        0

b+ 0x8049000 <_start> mov    %eax,%ecx
0x8049005 <_start+5> mov    %eax,%ebx
0x804900a <_start+10> mov    0x804a000,%ecx
0x804900f <_start+15> mov    %eax,%edx
0x8049014 <_start+20> int    0x80
> 0x8049016 <_start+22> mov    %eax,%ebx
0x804901b <_start+27> mov    %eax,%ecx
0x8049020 <_start+32> mov    0x804a008,%ecx
0x8049025 <_start+37> mov    %eax,%edx

native process 3962 in: _start
L14  PC: 0x8049016
eax    0x8      8
ecx    0x804a000  134520832
edx    0x8      8
ebx    0x1        1
esp    0xffffc330 0xffffc330
ebp    0x0        0
esi    0x0        0
edi    0x0        0
eip    0x8049016 0x8049016 <_start+22>
--Type <RET> for more, q to quit, c to continue without paging--

0x804a000 <msg1>: "Hello, "
Посмотрите значение переменной msg2 по адресу. Адрес переменной можно определить по декомпилированной инструкции. Посмотрите инструкцию mov    ecx,msg2 которая записывает в регистр ecx адрес переменной msg2 (рис. 9.4).

```

Рис. 2.8: -

Посмотрел значение переменной msg1 по имени и значение переменной msg2 по адресу(рис. [2.9])

```

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

native process 3962 In: _start
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0       0
gs      0x0       0
(gdb) x/1sb &msg1
0x804a000 <msg1>:    "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:    "world!\n\034"
(gdb)

```

Рис. 2.9: -

Изменил первый символ переменной msg1 (рис. [2.10])

```

0x804a008 <msg2>:    "world!\n\034"
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:    "hello, "
(gdb)

```

Рис. 2.10: -

Заменял первый символ во второй переменной msg2(рис. [2.11])

```

(gdb) set {char}&msg2='U'
(gdb) x/1sb &msg2
0x804a008 <msg2>:    "Uorld!\n\034"
(gdb)

```

Рис. 2.11: -

Вывел в различных форматах значение регистра edx(рис. [2.12])

```
(gdb) p/s $edx
$3 = 8
(gdb) p/x
$4 = 0x8
(gdb) p/t
$5 = 1000
(gdb) █
```

Рис. 2.12: -

С помощью команды set изменил значение регистра ebx(рис. [2.13])

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s
$7 = 50
(gdb) p/s $ebx
$8 = 2
(gdb) █
```

Рис. 2.13: -

Скопировал файл lab8-2.asm, создай исполняемый файл, загрузил исполняемый файл в отладчик, указав аргументы, создал точку останова на метке \_start и запусти программу, посмотрел на содержимое того, что расположено по адресу, шаг изменения адреса равен 4, потому что они располагаются в 4 байтах друг от друга, а столько занимает элемент стека (рис. [2.14])

```
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffc2e0: 0x00000005
(gdb) x/s *(void**)( $esp + 4)
0xffffc574: "/afs/.dk.sci.pfu.edu.ru/home/n/e/neardeev/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8)
0xffffc5b9: "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xffffc5cb: "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xffffc5dc: "2"
(gdb) x/s *(void**)( $esp + 20)
0xffffc5de: "аргумент 3\n\n\n\n\n"
(gdb) x/s *(void**)( $esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb) 
```

```
%include 'in_out.asm'

STARTUP 1---
```

Рис. 2.14: -

### **3 Задания для самостоятельной работы**

Программа из лабораторной 8 с использованием подпрограмм из 9-ой (рис. [3.1])

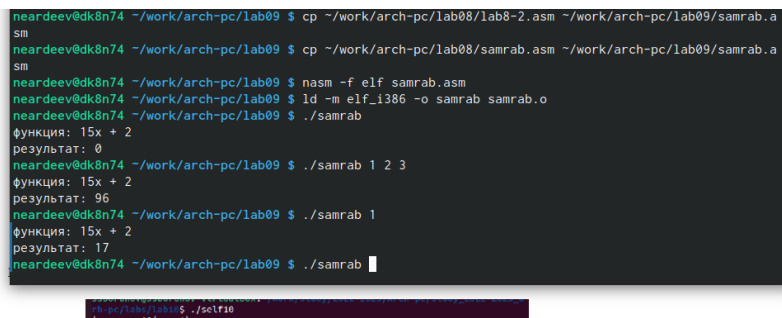
```

1 %include 'in_out.asm'
2
3 SECTION .data
4 f_x db "функция: 15x + 2",0h
5 msg db 10,13,'результат: ',0h
6
7 SECTION .text
8 global _start
9
10 _f:
11 push ebx
12 mov ebx, 15
13 mul ebx
14 add eax,2
15 pop ebx
16 ret
17
18 _start:
19 pop ecx
20 pop edx
21 sub ecx,1
22 mov esi, 0
23
24 next:
25 cmp ecx,0h
26 jz _end
27 pop eax
28 call atoi
29 call _f
30 add esi, eax
31
32 loop next
33
34 _end:

```

Рис. 3.1: -

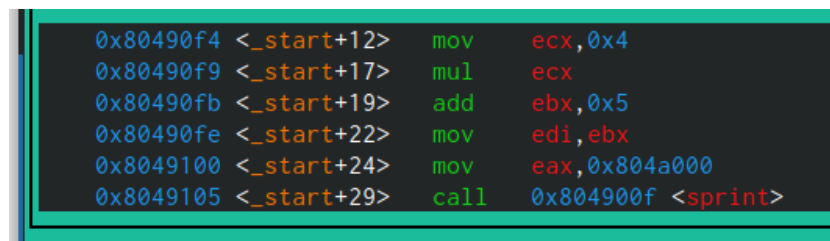
Ее работа(рис. [3.2])



```
neardeev@dk8n74 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/samrab.asm
neardeev@dk8n74 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/samrab.asm ~/work/arch-pc/lab09/samrab.asm
neardeev@dk8n74 ~/work/arch-pc/lab09 $ nasm -f elf samrab.asm
neardeev@dk8n74 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o samrab samrab.o
neardeev@dk8n74 ~/work/arch-pc/lab09 $ ./samrab
функция: 15x + 2
результат: 0
neardeev@dk8n74 ~/work/arch-pc/lab09 $ ./samrab 1 2 3
функция: 15x + 2
результат: 96
neardeev@dk8n74 ~/work/arch-pc/lab09 $ ./samrab 1
функция: 15x + 2
результат: 17
neardeev@dk8n74 ~/work/arch-pc/lab09 $ ./samrab
```

Рис. 3.2: -

Посмотрел регистры, чтобы найти ошибку (рис. [3.3])



```
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>
```

Рис. 3.3: -

Ошибка была в сторках

```
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
```

Работа правильной программы(рис. [3.4])

```
Reading symbols from samrab-1...  
(gdb) run  
Starting program: /afs/.dk.sci.pfu.edu.ru/home/n/e/neardeev/work/arch-pc/lab09/samrab-1  
Результат: 25  
[Inferior 1 (process 5443) exited normally]  
(gdb) █
```

Рис. 3.4: -



## 4 Выводы

В результате выполнения работы, я приобрел навыки написания программ с использованием подпрограмм и познакомился с базовыми функциями отладчика gdb.