

# **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №10**

**Дисциплина: Архитектура компьютера**

Обрезкова Анастасия Владимировна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Реализация подпрограмм в NASM . . . . .	9
4.2	Отладка программ с помощью GDB . . . . .	14
4.3	Добавление точек останова . . . . .	17
4.4	Работа с данными программы в GDB . . . . .	18
4.5	Обработка аргументов командной строки в GDB . . . . .	21
4.6	Задания для самостоятельной работы . . . . .	24
<b>5</b>	<b>Выводы</b>	<b>29</b>
	<b>Список литературы</b>	<b>30</b>

## Список иллюстраций

4.1	Создание, переход в lab09 . . . . .	9
4.2	Ввод текста . . . . .	10
4.3	Результат программы . . . . .	11
4.4	Изменения текста . . . . .	12
4.5	Изменения текста . . . . .	13
4.6	Результат изменений . . . . .	13
4.7	Текст программы . . . . .	14
4.8	Вывела результат . . . . .	15
4.9	Проверка работы . . . . .	15
4.10	Установка брейкпоинта . . . . .	15
4.11	Просмотр кода . . . . .	16
4.12	Переключение на отображение . . . . .	16
4.13	Режим псевдографики . . . . .	17
4.14	Проверка . . . . .	17
4.15	Точка останова . . . . .	18
4.16	Информация . . . . .	18
4.17	Содержимое регистров . . . . .	19
4.18	Значение переменной . . . . .	19
4.19	Замена символа . . . . .	19
4.20	Переменная msg2 . . . . .	20
4.21	Значение регистров . . . . .	20
4.22	Изменила значение регистра . . . . .	21
4.23	Копирование . . . . .	21
4.24	Исполняемый файл . . . . .	21
4.25	Ключ -args . . . . .	22
4.26	Установка точки останова . . . . .	22
4.27	. . . . .	23
4.28	Позиции стека . . . . .	23
4.29	Позиции стека . . . . .	24
4.30	Текст программы . . . . .	25
4.31	Текст программы . . . . .	26
4.32	Результат . . . . .	27
4.33	Начальный текст программы . . . . .	27
4.34	Результат начальной программы . . . . .	27
4.35	Измененная программа . . . . .	28
4.36	Результат . . . . .	28

## Список таблиц

# 1 Цель работы

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

## 2 Задание

1. Приобрести навыки написания программ с использованием подпрограмм.
2. Ознакомиться с методами отладки при помощи GDB и его основными возможностями.

### 3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

- обнаружение ошибки;
- поиск её местонахождения;
- определение причины ошибки;
- исправление ошибки.

Наиболее часто применяют следующие методы отладки:

- создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения);
- использование специальных программ-отладчиков.

Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам.

GDB (GNU Debugger — отладчик проекта GNU) [1] работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его

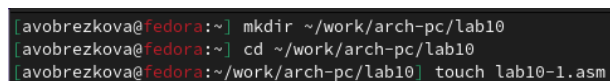
в качестве базовой подсистемы отладки.



## 4 Выполнение лабораторной работы

### 4.1 Реализация подпрограмм в NASM


1. Создала каталог для программ лабораторной работы №10, перешла в него и создала файл lab9=10-1.asm. (рис. 4.1)



```
[avobrezkova@fedora:~] mkdir ~/work/arch-pc/lab10  
[avobrezkova@fedora:~] cd ~/work/arch-pc/lab10  
[avobrezkova@fedora:~/work/arch-pc/lab10] touch lab10-1.asm
```

Рис. 4.1: Создание, переход в lab09

2. Ввела в файл lab10-1 нужный текст программы из листинга 10.1., создала исполняемый файл и вывела результат. (рис. 4.2; рис. 4.3)

Открыть ▾ 

• lab10-1.asm  
~/work/arch-pc/lab10

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0

SECTION .bss
x: RESB 80
rezs: 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 [rez],eax


ret
```

Рис. 4.2: Ввод текста

```
[avobrezkova@fedora:~/work/arch-pc/lab10] nasm -f elf lab10-1.asm
[avobrezkova@fedora:~/work/arch-pc/lab10] ld -m elf_i386 -o lab10-1 lab10-1.o
[avobrezkova@fedora:~/work/arch-pc/lab10] ./lab10-1
Введите x: 2
2x+7=11
[avobrezkova@fedora:~/work/arch-pc/lab10]
```

Рис. 4.3: Результат программы

3. Изменила текст программы, для вычисления выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x)=2x+7$ ,  $g(x)=3x-1$ . (рис. 4.4; рис. 4.5; рис. 4.6)

Открыть ▾  • lab10-1.asm  
~/work/arch-pc/lab10

```
%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
```

Рис. 4.4: Изменения текста

```

call _calcul

mov eax,result
call sprint
mov eax,[res]
call iprintLF

call quit

_calcul:

call _subcalcul

mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret

_subcalcul:
mov ebx,3
mul ebx
sub eax,1
ret

```

Рис. 4.5: Изменения текста

```

[avobrezkova@fedora:~/work/arch-pc/lab10] nasm -f elf lab10-1.asm
[avobrezkova@fedora:~/work/arch-pc/lab10] ld -m elf_i386 -o lab10-1 lab10-1.o
[avobrezkova@fedora:~/work/arch-pc/lab10] ./lab10-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 1
f(g(x))= 11
[avobrezkova@fedora:~/work/arch-pc/lab10] █

```

Рис. 4.6: Результат изменений

## 4.2 Отладка программ с помощью GDB

1. Создала файл lab10-2.asm с текстом программы из листинга 10.2. Получила исполняемый файл для работы с GDB и загрузила его в gdb.(рис. 4.7; рис. 4.8)



```
lab10-2.asm
~/work/arch-pc/lab10

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. 4.7: Текст программы

```
avobrezkova@fedora:~/work/arch-pc/lab10 — gdb lab10-2
[avobrezkova@fedora:~/work/arch-pc/lab10] nasm -f elf -g -l lab10-2.lst lab10-2.asm
[avobrezkova@fedora:~/work/arch-pc/lab10] ld -m elf_i386 -o lab10-2 lab10-2.o
[avobrezkova@fedora:~/work/arch-pc/lab10] gdb lab10-2
GNU gdb (GDB) Fedora 11.2-3.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)
```

Рис. 4.8: Вывела результат

2. Проверьте работу программы, запустив ее в оболочке GDB с помощью команды `run`. (рис. 4.9)

```
(gdb) run
Starting program: /home/avobrezkova/work/arch-pc/lab10/lab10-2
Hello, world!
[Inferior 1 (process 2884) exited normally]
(gdb)
```

Рис. 4.9: Проверка работы

3. Установила брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустила её. (рис. 4.10)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 12.
(gdb) run
Starting program: /home/avobrezkova/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:12
12      mov eax, 4
(gdb)
```

Рис. 4.10: Установка брейкпоинта

4. Посмотрела дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`. (рис. 4.11)

```
(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. 4.11: Просмотр кода

5. Переключидась на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`. (рис. 4.12)

```
(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. 4.12: Переключение на отображение

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel: в АТТ перед адресом регистра ставится \$, а перед названием регистра %, сначала записывается адрес, а потом - регистр. В Intel сначала регистр, а потом адрес, и перед ними ничего не ставится.

6. Включите режим псевдографики для более удобного анализа программы. (рис. 4.13)



```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0

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

native process 3076 In: _start L12 PC: 0x8049000
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) yStarting program: /home/avobrezkova/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:12
(gdb)

```

Рис. 4.13: Режим псевдографики

## 4.3 Добавление точек останова

1. Ранее я установила точку останова по имени метки (`_start`). Проверила это с помощью команды `info breakpoints` (кратко `i b`). (рис. 4.14)

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0

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

native process 3076 In: _start L12 PC: 0x8049000

Breakpoint 1, _start () at lab10-2.asm:12
(gdb) info breakpoints
Num      Type             Disp Enb Address      What
1        breakpoint       keep y  0x08049000 lab10-2.asm:12
          breakpoint already hit 1 time
(gdb)

```

Рис. 4.14: Проверка

2. Установила еще одну точку останова по адресу инструкции.. (рис. 4.15)

```

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al

native process 3076 In: _start L12 PC: 0x8049000
Num  Type      Disp Enb Address  What
1    breakpoint keep y  0x08049000 lab10-2.asm:12
    breakpoint already hit 1 time
(gdb) layout asm
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab10-2.asm, line 25.
(gdb)

```

Рис. 4.15: Точка останова

3. Посмотрела информацию о всех установленных точках останова. (рис. 4.16)

```

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al

native process 3076 In: _start L12 PC: 0x8049000
Breakpoint 2 at 0x8049031: file lab10-2.asm, line 25.
(gdb) i b
Num  Type      Disp Enb Address  What
1    breakpoint keep y  0x08049000 lab10-2.asm:12
    breakpoint already hit 1 time
2    breakpoint keep y  0x08049031 lab10-2.asm:25
(gdb)

```

Рис. 4.16: Информация

## 4.4 Работа с данными программы в GDB

1. Посмотрела содержимое регистров с помощью команды info registers. (рис. 4.17)

```

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al

native process 3076 In: _start                                L12  PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd200 0xffffd200
ebp      0x0      0
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 4.17: Содержимое регистров

2. Посмотрела значение переменной msg1. (рис. 4.18)

```

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al

native process 3076 In: _start                                L12  PC: 0x8049000
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) x/1sb &msg1
0x804a000 <msg1>:  "Hello, "
(gdb)

```

Рис. 4.18: Значение переменной

3. Изменила первый символ переменной msg1. (рис. 4.19)

```

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al

native process 3076 In: _start                                L12  PC: 0x8049000
fs       0x0      0
gs       0x0      0
(gdb) x/1sb &msg1
0x804a000 <msg1>:  "Hello, "
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:  "hello, "
(gdb)

```

Рис. 4.19: Замена символа

4. Посмотрела значение переменной `msg2` по адресу и изменила первый символ переменной `msg1`. (рис. 4.20)

```
0x8049010 <_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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al

native process 3076 In: _start                               L12  PC: 0x8049000
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}msg2='h'
'msg2' has unknown type; cast it to its declared type
(gdb) set {char}&msg2='q'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "qorld!\n\034"
(gdb)
```

Рис. 4.20: Переменная `msg2`

5. Посмотрела значения регистров с помощью команды `print /F`. (рис. 4.21)

```
0x8049010 <_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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al

native process 3076 In: _start                               L12  PC: 0x8049000
0x804a008 <msg2>:      "qorld!\n\034"
(gdb) p/x $edx
$1 = 0x0
(gdb) p/t $edx
$2 = 0
(gdb) p/s $edx
$3 = 0
(gdb)
```

Рис. 4.21: Значение регистров

6. С помощью команды `set` изменила значение регистра `ebx`. (рис. 4.22)

```

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al

native process 3076 In: _start                               L12  PC: 0x8049000
$3 = 0
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb)

```

Рис. 4.22: Изменила значение регистра

## 4.5 Обработка аргументов командной строки в GDB

1. Скопировала файл lab9-2.asm, созданный при выполнении лабораторной работы No9, с программой выводящей на экран аргументы командной строки (Листинг 9.2) в файл с именем lab10-3.asm. (рис. 4.23)

```

[avobrezkova@fedora:~] cp ~/work/arch-pc/lab09/lab9-2.asm ~/work/arch-pc/lab10/lab10-3.asm

```

Рис. 4.23: Копирование

2. Создала исполняемый файл. (рис. 4.24)

```

[avobrezkova@fedora:~/work/arch-pc/lab10] nasm -f elf -g -l lab10-3.lst lab10-3.asm
[avobrezkova@fedora:~/work/arch-pc/lab10] ld -m elf_i386 -o lab10-3 lab10-3.o

```

Рис. 4.24: Исполняемый файл

3. Использовала ключ `-args`. Загрузила исполняемый файл в отладчик, указав аргументы. (рис. 4.25)

```

[avobrezkova@fedora:~/work/arch-pc/lab10] gdb --args lab10-3 аргумент1 аргумент
2 'аргумент 3'
GNU gdb (GDB) Fedora 11.2-3.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...

```

Рис. 4.25: Ключ `–args`

4. Установила точку останова перед первой инструкцией в программе и запустила ее. (рис. 4.26)

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 7.
(gdb) run
Starting program: /home/avobrezkova/work/arch-pc/lab10/lab10-3 аргумент1 аргумент
2 аргумент\ 3

```

Рис. 4.26: Установка точки останова

5. Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). (рис. 4.27)

```

avobrezkova@fedora:~/work/arch-pc/lab10 — gdb --args lab10...
Register group: general
eax      0x0      0
ecx      0x5      5
edx      0x0      0
ebx      0x0      0
esp      0xffffd1c4 0xffffd1c4
ebp      0x0      0x0
esi      0x0      0

B+  7  pop ecx
    8
    > 9  pop edx
    10
    11 sub ecx, 1
    12
    13 next:
    14 cmp ecx, 0

native process 4080 In: _start L9 PC: 0x80490e9
(gdb) n
(gdb) x/x $esp
0xffffd1c4: 0xffffd374
(gdb)

```

Рис. 4.27: .

6. Посмотрела остальные позиции стека. (рис. 4.28; рис. 4.29)

```

avobrezkova@fedora:~/work/arch-pc/lab10 — gdb --args lab10...
Register group: general
eax      0x0      0
ecx      0x5      5
edx      0x0      0
ebx      0x0      0
esp      0xffffd1c4 0xffffd1c4
ebp      0x0      0x0
esi      0x0      0

lab10-3.asm
13  next:
14  cmp ecx, 0
15  jz _end
16
17  pop eax
18  call sprintf
19  loop next
20

native process 4080 In: _start L9 PC: 0x80490e9
(gdb) x/x $esp
0xffffd1c4: 0xffffd374
(gdb) x/s *(void*)($esp + 4)
0xffffd3a1: "аргумент1"
(gdb) x/s *(void*)&esp + 8)
No symbol "esp" in current context.
(gdb) x/s *(void*)($esp + 8)
0xffffd3b3: "аргумент"
(gdb)

```

Рис. 4.28: Позиции стека

```
avobrezkova@fedora:~/work/arch-pc/lab10 — gdb --args lab10...
Register group: general
eax      0x0      0
ecx      0x5      5
edx      0x0      0
ebx      0x0      0
esp      0xffffd1c4 0xffffd1c4
ebp      0x0      0x0
esi      0x0      0

lab10-3.asm
13 next:
14 cmp ecx, 0
15 jz _end
16
17 pop eax
18 call sprintf
19 loop next
20

native process 4080 In: _start L9 PC: 0x80490e9
(gdb) x/s *(void**)(esp + 8)
0xffffd3b3: "аргумент"
(gdb) x/s *(void**)(esp + 12)
0xffffd3cd: "2"
(gdb) x/s *(void**)(esp + 16)
0xffffd3cc: "аргумент 3"
(gdb) x/s *(void**)(esp + 20)
0x0: <error: Cannot access memory at address 0x0>
(gdb) |
```


Рис. 4.29: Позиции стека

Шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] и т.д.), потому что в теле цикла 4 строки кода.

## 4.6 Задания для самостоятельной работы

1. Преобразовала программу из лабораторной работы No9 (Задание No1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму. (рис. 4.30; рис. 4.31; рис. 4.32)



Открыть ▾ 

samrab10-1.asm  
~/work/arch-pc/lab10

```
%include 'in_out.asm'

SECTION .data
prim DB 'f(x) = 7 + 2x',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 kotik

add esi, eax
loop next

_end:
mov eax,otv
call sprint
mov eax,esi
call iprintLF
call quit
```

Рис. 4.30: Текст программы

Открыть ▾ 

samrab10-1.asm  
~/work/arch-pc/lab10

```
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 kotik

add esi, eax
loop next

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

kotik:
call atoi
mov ebx,2
mul ebx
add eax,7
ret
```

Рис. 4.31: Текст программы

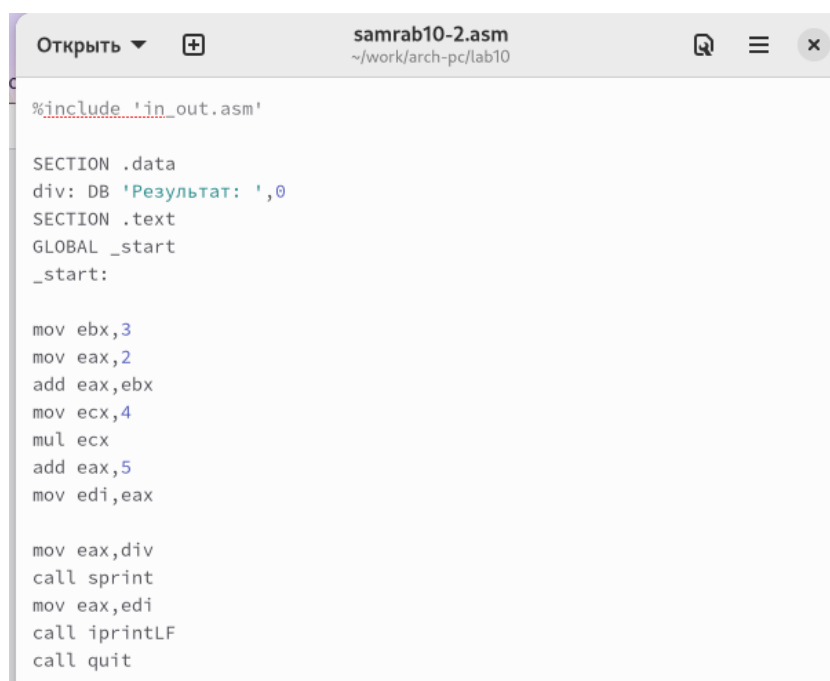
```

avobrezkova@fedora:~/work/arch-pc/lab10] nasm -f elf samrab10-1.asm
avobrezkova@fedora:~/work/arch-pc/lab10] ld -m elf_i386 -o samrab10-1 samrab10-1.o
avobrezkova@fedora:~/work/arch-pc/lab10] ./samrab10-1
f(x) = 7 + 2x
Результат: 0
avobrezkova@fedora:~/work/arch-pc/lab10] ./samrab10-1 1 2 3
f(x) = 7 + 2x
Результат: 33
avobrezkova@fedora:~/work/arch-pc/lab10]

```

Рис. 4.32: Результат

2. В листинге 10.3 приведена программа вычисления выражения  $(3 + 2) \times 4 + 5$ . При запуске данная программа дает неверный результат. Проверила это с помощью отладчика GDB, анализируя изменения значений регистров, определила ошибку и исправила ее. (рис. 4.33; рис. 4.34; рис. 4.35; рис. 4.36)



```

%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

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

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

```

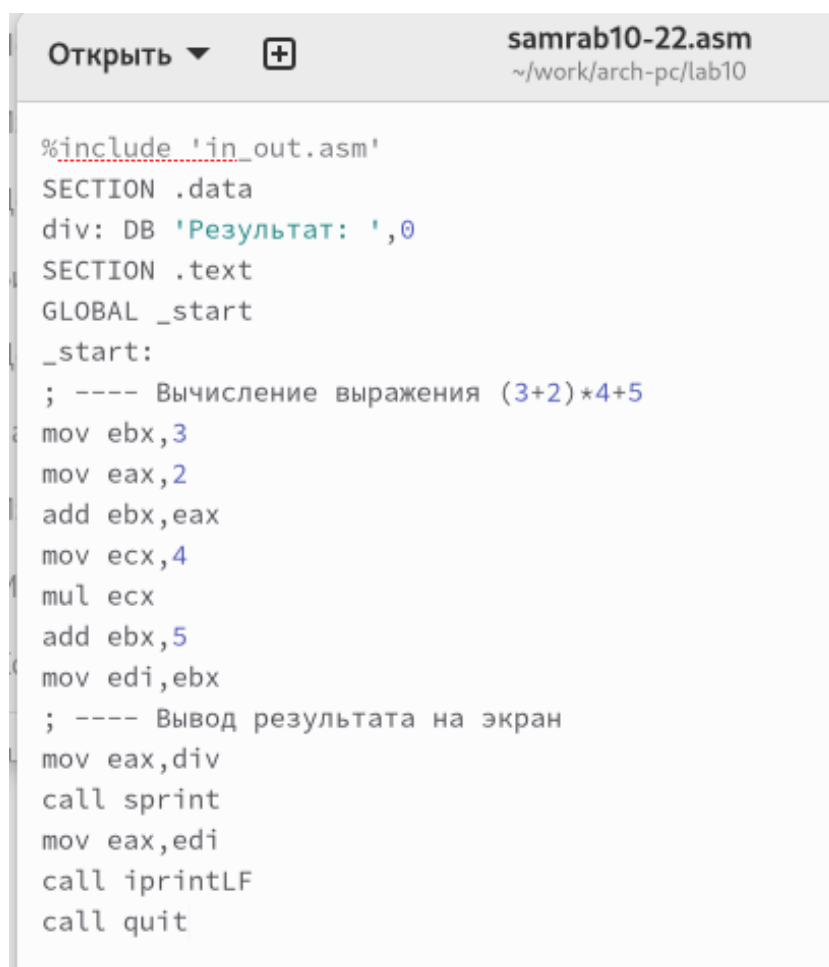
Рис. 4.33: Начальный текст программы

```

avobrezkova@fedora:~/work/arch-pc/lab10] nasm -f elf samrab10-2.asm
avobrezkova@fedora:~/work/arch-pc/lab10] ld -m elf_i386 -o samrab10-2 samrab10-2.o
avobrezkova@fedora:~/work/arch-pc/lab10] ./samrab10-2
Результат: 25
avobrezkova@fedora:~/work/arch-pc/lab10]

```

Рис. 4.34: Результат начальной программы




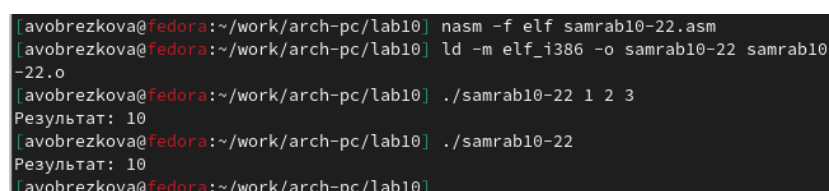
```
Открыть ▾  samrab10-22.asm  
~/work/arch-pc/lab10  
  
%include 'in_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
SECTION .text  
GLOBAL _start  
_start:  
; ---- Вычисление выражения (3+2)*4+5  
mov ebx,3  
mov eax,2  
add ebx,eax  
mov ecx,4  
mul ecx  
add ebx,5  
mov edi,ebx  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit
```

Рис. 4.35: Измененная программа



```
[avobrezkova@fedora:~/work/arch-pc/lab10] nasm -f elf samrab10-22.asm  
[avobrezkova@fedora:~/work/arch-pc/lab10] ld -m elf_i386 -o samrab10-22 samrab10-22.o  
[avobrezkova@fedora:~/work/arch-pc/lab10] ./samrab10-22 1 2 3  
Результат: 10  
[avobrezkova@fedora:~/work/arch-pc/lab10] ./samrab10-22  
Результат: 10  
[avobrezkova@fedora:~/work/arch-pc/lab10]
```

Рис. 4.36: Результат

Данные изменения можно проверить по ссылке: [https://github.com/avobrezkova/study\\_2022-2023\\_arh-pc/tree/master/labs/lab10](https://github.com/avobrezkova/study_2022-2023_arh-pc/tree/master/labs/lab10)

## 5 Выводы

Приобрела навыки написания программ с использованием подпрограмм. Ознакомилась с методами отладки при помощи GDB и его основными возможностями.

## Список литературы

1. [https://esystem.rudn.ru/pluginfile.php/1584394/mod\\_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%9610.pdf](https://esystem.rudn.ru/pluginfile.php/1584394/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%9610.pdf)