

# **Лабораторная работа №9**

**Понятие подпрограммы**

Самарханова Полина Тимуровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Реализация подпрограмм в NASM . . . . .	7
3.2	Отладка программ с помощью GDB . . . . .	11
3.3	Добавление точек останова . . . . .	15
3.4	Работа с данными программы в GDB . . . . .	16
3.5	Обработка аргументов командной строки в GDB . . . . .	19
3.6	Задание для самостоятельной работы . . . . .	20
<b>4</b>	<b>Выводы</b>	<b>26</b>
	<b>Список литературы</b>	<b>27</b>

# Список иллюстраций

3.1	Создание каталога . . . . .	7
3.2	Перемещение по директории . . . . .	7
3.3	Создание файла . . . . .	7
3.4	Копирование файла . . . . .	7
3.5	Редактирование файла . . . . .	8
3.6	Запуск программы . . . . .	9
3.7	Редактирование файла . . . . .	10
3.8	Запуск программы . . . . .	11
3.9	Создание файла . . . . .	11
3.10	Редактирование файла . . . . .	12
3.11	Запуск исполняемого файла . . . . .	12
3.12	Запуск программы в отладчике . . . . .	13
3.13	Установка брейкпоинта . . . . .	13
3.14	Запуск . . . . .	13
3.15	Диссассимилированный код программы . . . . .	14
3.16	Отображение с Intel'овским синтаксисом . . . . .	15
3.17	Точка останова . . . . .	15
3.18	Установка точки останова . . . . .	16
3.19	Точки останова . . . . .	16
3.20	info register . . . . .	16
3.21	Значение переменной по имени . . . . .	17
3.22	Значение переменной по адресу . . . . .	17
3.23	Изменение переменной . . . . .	17
3.24	Изменение второй переменной . . . . .	17
3.25	Изменение значений в разные форматы . . . . .	18
3.26	Изменение значений ebx . . . . .	18
3.27	Копирование файла . . . . .	19
3.28	Создание файла . . . . .	19
3.29	Запуск программы с точкой останова . . . . .	19
3.30	Регистр esp . . . . .	20
3.31	Позиции стека . . . . .	20
3.32	Создание файла . . . . .	20
3.33	Редактирование файла . . . . .	21
3.34	Запуск программы . . . . .	21
3.35	Редактирование файла . . . . .	22
3.36	Запуск программы в отладчике . . . . .	23
3.37	Действия в отладчике . . . . .	24

3.38	Измененная программа . . . . .	25
3.39	Запуск программы . . . . .	25

# 1 Цель работы

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

## 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Задание для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация подпрограмм в NASM

Я создала каталог для выполнения работы №9 (рис. 3.1).

```
[spolina@fedora ~]$ mkdir ~/work/arch-pc/lab09
```

Рис. 3.1: Создание каталога

После перешла в созданную директорию (рис. 3.2).

```
[spolina@fedora ~]$ cd ~/work/arch-pc/lab09
```

Рис. 3.2: Перемещение по директории

Создала файл lab09-1.asm в новом каталоге (рис. 3.3).

```
[spolina@fedora lab09]$ touch lab09-1.asm
```

Рис. 3.3: Создание файла

Далее я скопировала файл in\_out.asm в созданный каталог(рис. 3.4).

```
[spolina@fedora lab09]$ cp ~/Загрузки/in_out.asm in_out.asm  
[spolina@fedora lab09]$ ls  
in_out.asm  lab09-1.asm
```

Рис. 3.4: Копирование файла

После чего я открыла файл в GNU nano и переписала код программы из листинга 9.1 (рис. 3.5).

```
GNU nano 7.2
#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:
;-----
; Основная программа
;-----
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. 3.5: Редактирование файла



Далее создала объектный файл программы и после компоновки запустила его (рис. 3.6).

```
[spolina@fedora lab09]$ nasm -f elf lab09-1.asm
[spolina@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[spolina@fedora lab09]$ ./lab09-1
Введите x: 5
2x+7=17
```

Рис. 3.6: Запуск программы

Я изменила текст файла, добавив подпрограмму `sub_calcul` в подпрограмму `_calcul` (рис. 3.7).

```

GNU nano 7.2
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+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
_calcul:
call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx,3
mul ebx
sub eax,1
ret

```

Рис. 3.7: Редактирование файла

После я запустила исполняемый файл (рис. 3.8).

```
[spolina@fedora lab09]$ nasm -f elf lab09-1.asm
[spolina@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[spolina@fedora lab09]$ ./lab09-1
Введите x: 5
 $2(3x-1)+7=35$ 
```

Рис. 3.8: Запуск программы

## 3.2 Отладка программ с помощью GDB

Далее я создала файл lab09-2.asm, используя команду touch (рис. 3.9).

```
[spolina@fedora lab09]$ touch lab09-2.asm
```

Рис. 3.9: Создание файла

Записала код программы из листинга 9.2, который выводит сообщение Hello world (рис. 3.10).

```

GNU nano 7.2
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. 3.10: Редактирование файла

Получила исполняемый файл. Для работы с GDB провела трансляцию программ с ключом “-g” и загрузила исполняемый файл в отладчик (рис. 3.11).

```

[spolina@fedora lab09]$ nasm -f elf -g -l lab090-2.lst lab090-2.asm
[spolina@fedora lab09]$ ld -m elf_i386 -o lab090-2 lab090-2.o
[spolina@fedora lab09]$ gdb lab090-2

```

Рис. 3.11: Запуск исполняемого файла

После я проверила работу программы в оболочке GDB с помощью команды run

(рис. 3.12).

```
GNU gdb (GDB) Fedora Linux 13.2-5.fc38
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-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 lab090-2...
(gdb) run
Starting program: /home/spolina/work/arch-pc/lab09/lab090-2

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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4078) exited normally]
(gdb)
```

Рис. 3.12: Запуск программы в отладчике

Установила брейкпоинт на метку `_start`, с которой начинается выполнение ассемблерной программы (рис. 3.13).

```
(gdb) break _start
Breakpoint 1 at 0x4010e0: file lab090-2.asm, line 9.
```

Рис. 3.13: Установка брейкпоинта

Далее запустила её (рис. 3.14).

```
(gdb) run
Starting program: /home/spolina/work/arch-pc/lab09/lab090-2

Breakpoint 1, _start () at lab090-2.asm:9
9      mov eax, 4
```

Рис. 3.14: Запуск

С помощью команды “disassemble \_start” я просмотрела дисассимилированный код программы (рис. 3.15).

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x004010e0 <+0>:      mov     $0x4,%eax
    0x004010e5 <+5>:      mov     $0x1,%ebx
    0x004010ea <+10>:     mov     $0x402118,%ecx
    0x004010ef <+15>:     mov     $0x8,%edx
    0x004010f4 <+20>:     int     $0x80
    0x004010f6 <+22>:     mov     $0x4,%eax
    0x004010fb <+27>:     mov     $0x1,%ebx
    0x00401100 <+32>:     mov     $0x402120,%ecx
    0x00401105 <+37>:     mov     $0x7,%edx
    0x0040110a <+42>:     int     $0x80
    0x0040110c <+44>:     mov     $0x1,%eax
    0x00401111 <+49>:     mov     $0x0,%ebx
    0x00401116 <+54>:     int     $0x80
End of assembler dump.
```

Рис. 3.15: Дисассимилированный код программы

После я переключилась на отображение команд с Intel’овским синтаксисом, введя команду “set disassembly-flavor intel” (рис. 3.16).

```

(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x004010e0 <+0>:      mov     eax,0x4
    0x004010e5 <+5>:      mov     ebx,0x1
    0x004010ea <+10>:     mov     ecx,0x402118
    0x004010ef <+15>:     mov     edx,0x8
    0x004010f4 <+20>:     int     0x80
    0x004010f6 <+22>:     mov     eax,0x4
    0x004010fb <+27>:     mov     ebx,0x1
    0x00401100 <+32>:     mov     ecx,0x402120
    0x00401105 <+37>:     mov     edx,0x7
    0x0040110a <+42>:     int     0x80
    0x0040110c <+44>:     mov     eax,0x1
    0x00401111 <+49>:     mov     ebx,0x0
    0x00401116 <+54>:     int     0x80
End of assembler dump.

```

Рис. 3.16: Отображение с Intel'овским синтаксисом

Основное различие заключается в том, что в режиме Intel пишется сначала сама команда, а потом её машинный код, в то время как в режиме AT&T идет сначала машинный код, а только потом сама команда.

### 3.3 Добавление точек останова

Я проверила наличие точки останова с помощью команды `info breakpoints` (рис. 3.17).

```

(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x004010e0 lab090-2.asm:9
breakpoint already hit 1 time

```

Рис. 3.17: Точка останова

Установила ещё одну точку останова по адресу инструкции (рис. 3.18).

```
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031
```

Рис. 3.18: Установка точки останова

Просмотрела информацию (рис. 3.19).

```
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y 0x004010e0 lab090-2.asm:9
          breakpoint already hit 1 time
2        breakpoint     keep y 0x08049031
```

Рис. 3.19: Точки останова

### 3.4 Работа с данными программы в GDB

Далее я просмотрела содержимое регистров с помощью команды `info register` (рис. 3.20).

```
(gdb) i r
eax                0x0                0
ecx                0x0                0
edx                0x0                0
ebx                0x0                0
esp                0xffffd1d0         0xffffd1d0
ebp                0x0                0x0
esi                0x0                0
edi                0x0                0
eip                0x4010e0           0x4010e0 <_start>
eflags             0x202             [ IF ]
cs                 0x23              35
ss                 0x2b              43
ds                 0x2b              43
es                 0x2b              43
fs                 0x0                0
gs                 0x0                0
```

Рис. 3.20: `info register`



Узнала значение переменной msg1 по имени (рис. 3.21).

```
(gdb) x/1sb &msg1
0x402118 <msg1>:      "Hello, "
```

Рис. 3.21: Значение переменной по имени

После просмотрела значение переменной msg2 по адресу, который можно определить по дизассемблированной инструкции (рис. 3.22).

```
(gdb) x/1sb 0x402120
0x402120 <msg2>:      "world!\n\034"
```

Рис. 3.22: Значение переменной по адресу

Изменила первый символ переменной msg1 (рис. 3.23).

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x402118 <msg1>:      "hello, "
```

Рис. 3.23: Изменение переменной

Также поменяла первый символ переменной msg2 (рис. 3.24).

```
(gdb) set {char}&msg2='L'
(gdb) x/1sb &msg2
0x402120 <msg2>:      "Lorld!\n\034"
```

Рис. 3.24: Изменение второй переменной

Далее я вывела значение регистра edx (рис. 3.25).

```

(gdb) p/s $edx
$1 = 7

(gdb) p/t $edx
$2 = 111

(gdb) p/x $edx
$3 = 0x7

```

Рис. 3.25: Изменение значений в разные форматы

С помощью команды set я изменила значение регистра ebx (рис. 3.26).

```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$3 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$4 = 2

```

Рис. 3.26: Изменение значений ebx

Значение регистра отличаются, так как в первом случае мы выводим код символа 2, который в десятичной системе счисления равен 50, а во втором случае выводится число 2, представленное в этой же системе.

## 3.5 Обработка аргументов командной строки в GDB

Я скопировала файл lab8-2.asm в файл с именем lab09-3.asm (рис. 3.27).

```
[spolina@fedora lab09]$ cp ~/work/study/2023-2024/Архитектура_компьютера/arch-pc/labs/lab08/report/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
```

Рис. 3.27: Копирование файла

Создала исполняемый файл,используя ключ `-args` для загрузки программы в GDB. Загрузила исполняемый файл,указав аргументы (рис. 3.28).

```
[spolina@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
[spolina@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
[spolina@fedora lab09]$ gdb --args lab09-3 аргумент1 аргумент2 'аргумент3'
GNU gdb (GDB) Fedora Linux 13.2-5.fc38
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-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 lab09-3...
```

Рис. 3.28: Создание файла

Установила точку останова перед первой инструкцией в программе и запустила её (рис. 3.29).

```
(gdb) b _start
Breakpoint 1 at 0x4011a8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/spolina/work/arch-pc/lab09/lab09-3 аргумент1 аргумент2 аргумент3

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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
```

Рис. 3.29: Запуск программы с точкой останова

Просмотрела адрес вершины стека, который хранится в регистре esp (рис. 3.30).

```
(gdb) x/x $esp
0xffffd190:      0x00000004
```

Рис. 3.30: Регистр esp

Ввела другие позиции стека(рис. 3.31).

```
(gdb) x/x $esp
0xffffd190:      0x00000004
(gdb) x/s *(void**)(esp + 4)
0xffffd341:      "/home/spolina/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd36a:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd37c:      "аргумент2"
(gdb) x/s *(void**)(esp + 16)
0xffffd38e:      "аргумент3"
(gdb) x/s *(void**)(esp + 22)
0xd3a00000:      <error: Cannot access memory at address 0xd3a00000>
(gdb)
```

Рис. 3.31: Позиции стека

Количество аргументов командной строки 4, следовательно и шаг равен четырем.

## 3.6 Задание для самостоятельной работы

Далее я создала файл для первого самостоятельного задания lab09-4.asm (рис. 3.32).

```
[2] ~ % cat lab09-4.asm
[spolina@fedora lab09]$ touch lab09-4.asm
[spolina@fedora lab09]$
```

Рис. 3.32: Создание файла

После чего отредактировала код программы lab8-4.asm, добавив подпрограмму, которая вычисляет значения функции  $f(x)$  (рис. 3.33).

```

GNU nano 7.2
#include 'in_out.asm'
SECTION .data
msg db "Ответ: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
call _calcul
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
_calcul:
mov ebx,10
mul ebx
sub eax,4
ret

```

Рис. 3.33: Редактирование файла

Далее я создала исполняемый файл и ввела аргументы (рис. 3.34).

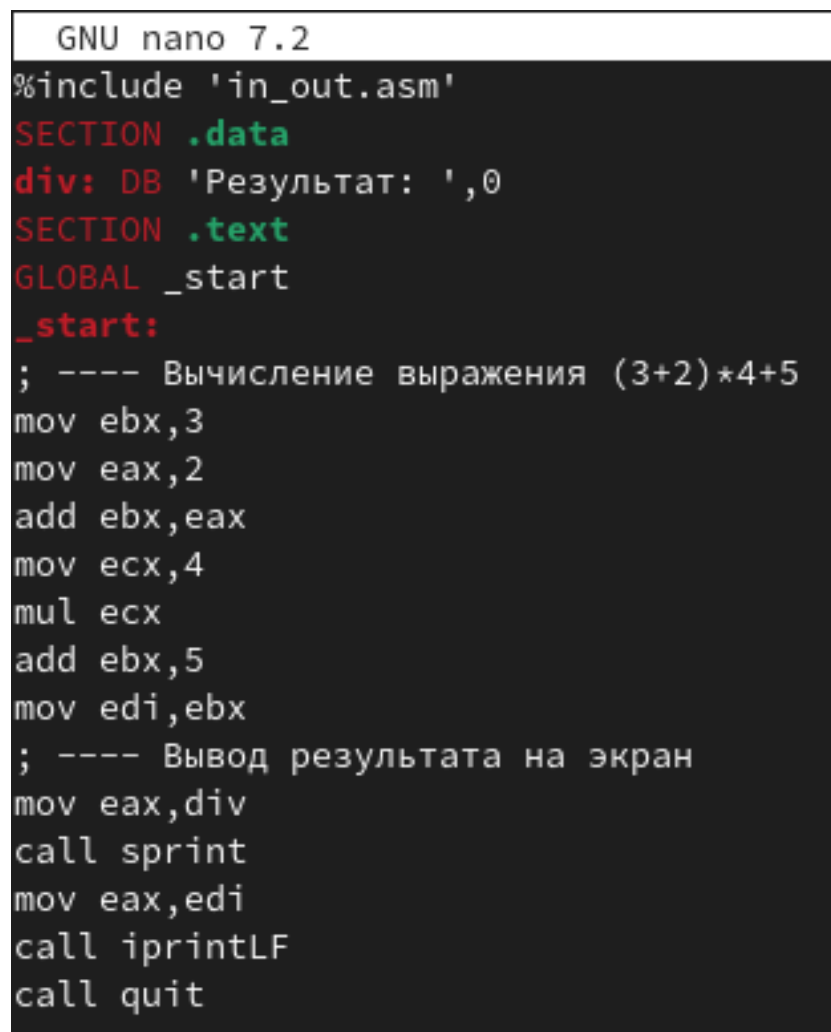
```

[spolina@fedora lab09]$ nasm -f elf lab09-4.asm
[spolina@fedora lab09]$ ld -m elf_i386 -o lab09-4 lab09-4.o
[spolina@fedora lab09]$ ./lab09-4 2 3 6 7
Ответ: 164

```

Рис. 3.34: Запуск программы

Создала файл и ввела код из листинга 9.3 (рис. 3.35).



```
GNU nano 7.2
#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. 3.35: Редактирование файла

Далее я открыла файл в отладчике GDB и запускаю программу (рис. 3.36).

```

[spolina@fedora lab09]$ nasm -f elf -g -l labl9.3.lst labl9.3.asm
[spolina@fedora lab09]$ ld -m elf_i386 -o labl9.3 labl9.3.o
[spolina@fedora lab09]$ gdb labl9.3
GNU gdb (GDB) Fedora Linux 13.2-5.fc38
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-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 labl9.3...
(gdb) run
Starting program: /home/spolina/work/arch-pc/lab09/labl9.3

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.
Результат: 10
[Inferior 1 (process 24174) exited normally]

```

Рис. 3.36: Запуск программы в отладчике

Просмотрев дисассимилированный код программы, поставила точку останова перед прибавлением 5 и открыла значения регистров на данном этапе (рис. 3.37).

```

(gdb) disassembler _start
Undefined command: "disassembler". Try "help".
(gdb) disassemble _start
Dump of assembler code for function _start:
    0x004011c8 <+0>:      mov     $0x3,%ebx
    0x004011cd <+5>:      mov     $0x2,%eax
    0x004011d2 <+10>:     add     %eax,%ebx
    0x004011d4 <+12>:     mov     $0x4,%ecx
    0x004011d9 <+17>:     mul     %ecx
    0x004011db <+19>:     add     $0x5,%ebx
    0x004011de <+22>:     mov     %ebx,%edi
    0x004011e0 <+24>:     mov     $0x4021f8,%eax
    0x004011e5 <+29>:     call    0x4010ef <sprint>
    0x004011ea <+34>:     mov     %edi,%eax
    0x004011ec <+36>:     call    0x401166 <iprintf>
    0x004011f1 <+41>:     call    0x4011bb <quit>
End of assembler dump.
(gdb) b *0x004011db
Breakpoint 1 at 0x4011db: file labl9.3.asm, line 13.
(gdb) run
Starting program: /home/spolina/work/arch-pc/lab09/labl9.3

Breakpoint 1, _start () at labl9.3.asm:13
13      add ebx,5
(gdb) i r
eax                0x8                8
ecx                0x4                4
edx                0x0                0
ebx                0x5                5
esp                0xffffd1d0         0xffffd1d0
ebp                0x0                0x0
esi                0x0                0
edi                0x0                0
eip                0x4011db           0x4011db <_start+19>
eflags            0x202              [ IF ]
cs                0x23               35
ss                0x2b               43
ds                0x2b               43
es                0x2b               43
fs                0x0                0
gs                0x0                0

```

Рис. 3.37: Действия в отладчике

Как можно увидеть, регистр `ecx` со значением 4 умножается не на `ebx`, сложенным с `eax`, а только с `eax` со значением 2. Значит нужно поменять значения регистров (например присвоить `eax` значение 3 и просто прибавит 2. После изменений программа будет выглядеть следующим образом: (рис. 3.38).



```

GNU nano 7.2
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov eax,3
mov ebx,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

```

Рис. 3.38: Измененная программа

Далее я запустила программу (рис. 3.39).

```

[spolina@fedora lab09]$ nasm -f elf labl9.3.asm
[spolina@fedora lab09]$ ld -m elf_i386 -o labl9.3 labl9.3.o
[spolina@fedora lab09]$ ./labl9.3
Результат: 25

```

Рис. 3.39: Запуск программы

## 4 Выводы

В данной работе я приобрела навыки написания программ с подпрограммами и познакомилась с методами отладки при помощи GDB.

# Список литературы

Лабораторная работа №9