

# **Отчёт по лабораторной работе №9**

**Выполнил студент НКАбд-02-25**

Арина Андреевна Дрекина

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Цель работы.</b>                                 | <b>3</b>  |
| <b>2</b> | <b>Порядок выполнения лабораторной работы.</b>      | <b>4</b>  |
| <b>3</b> | <b>Реализация подпрограмм в NASM.</b>               | <b>5</b>  |
| <b>4</b> | <b>Отладка программ с помощью GDB.</b>              | <b>11</b> |
| <b>5</b> | <b>Добавление точек останова.</b>                   | <b>18</b> |
| <b>6</b> | <b>Работа с данными программы в GDB.</b>            | <b>20</b> |
| <b>7</b> | <b>Обработка аргументов командной строки в GDB.</b> | <b>26</b> |
| <b>8</b> | <b>Задание для самостоятельной работы.</b>          | <b>30</b> |
| <b>9</b> | <b>Вывод.</b>                                       | <b>34</b> |

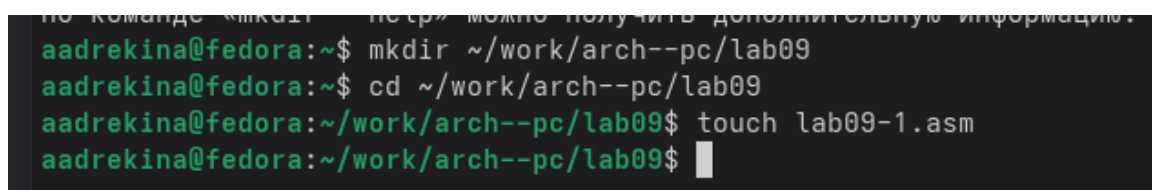
# **1 Цель работы.**

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

## **2 Порядок выполнения лабораторной работы.**

### 3 Реализация подпрограмм в NASM.

Я создала каталог для выполнения лабораторной работы №9, затем перешла в него и создала текстовый файл lab09-1.asm. (Рисунок 3.1)



```
по команде «mkdir -p» можно получить дополнительную информацию.  
aadrekina@fedora:~$ mkdir ~/work/arch--pc/lab09  
aadrekina@fedora:~$ cd ~/work/arch--pc/lab09  
aadrekina@fedora:~/work/arch--pc/lab09$ touch lab09-1.asm  
aadrekina@fedora:~/work/arch--pc/lab09$
```

Рисунок 3.1: Создание каталога для лабораторной работы.

Я ввела в файл текст из Листинга 9.1. Перед этим изучила программу (Рисунок 3.2)

```
%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
```

Рисунок 3.2: Ввод программы в файл.

Потом я запустила программу. (Рисунок 3.3)

```
aadrekina@fedora:~/work/arch--pc/lab09$ nasm -f elf lab09-1.asm
aadrekina@fedora:~/work/arch--pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
aadrekina@fedora:~/work/arch--pc/lab09$ ./lab09-1
Введите x: 5
2x+7=17
aadrekina@fedora:~/work/arch--pc/lab09$
```

Рисунок 3.3: Запуск.

Листинг 9.1:

```
%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

```

Потом я изменила текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, чтобы вычислялось выражение  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x)=2x+7$ ,  $g(x)=3x-1$ . (Рисунок 3.4)



11, 4 декабря 19:07

Открыть ▾ 

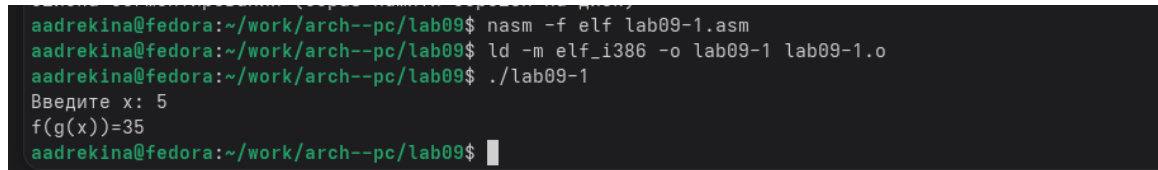
lab09-1.asm  
~/work/arch--pc/lab09

cmd | COURSE | Makefile | ЛО4\_Дрекина\_ | ЛО7\_Дреки

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB 'f(g(x))=',0
SECTION .bss
x: RESB 80
res: RESD 1
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
_calcul:
call _subcalcul
mov ebx, 2
xor edx, edx
mul ebx
add eax, 7
mov [res], eax
ret
_subcalcul:
mov ebx, 3
xor edx, edx
mul ebx
sub eax, 1
ret
```

Рисунок 3.4: Изменения в программе.

Затем я сделала файл исполняемым и запустила программу. (Рисунок 3.5)

A terminal window with a dark background and green text. The prompt is 'aadrekina@fedora:~/work/arch--pc/lab09\$'. The user enters 'nasm -f elf lab09-1.asm', then 'ld -m elf\_i386 -o lab09-1 lab09-1.o', and finally './lab09-1'. The program outputs 'Введите x: 5' and 'f(g(x))=35'.

```
aadrekina@fedora:~/work/arch--pc/lab09$ nasm -f elf lab09-1.asm
aadrekina@fedora:~/work/arch--pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
aadrekina@fedora:~/work/arch--pc/lab09$ ./lab09-1
Введите x: 5
f(g(x))=35
aadrekina@fedora:~/work/arch--pc/lab09$
```

Рисунок 3.5: Запуск программы.

Программа посчитала все правильно и я продолжила работу дальше.

## **4 Отладка программ с помощью GDB.**

Затем я создала еще один текстовый файл и ввела текст из Листинга 9.2. (Рисунок 4.1)

41, 4 декабря 19:29

Открыть ▾ +

lab09-2.asm  
~/work/arch--pc/lab09

RSE | Makefile | ЛО4\_Дрекина\_ | ЛО7\_Дрекина\_ | ЛО8\_Дрекина\_

```
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
```

Рисунок 4.1: Ввод текста из Листинга 9.2.

Листинг 9.2:

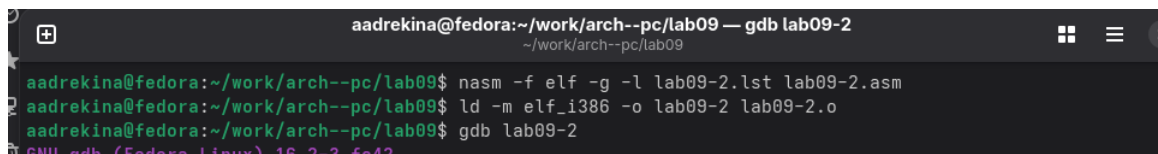
```
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
```

```

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

```

Потом я сделала этот файл исполняемым, но для работы с GDB в исполняемый файл я добавила отладочную информацию, для этого трансляцию программы я провела с ключом «-g». (Рисунок 4.2)



```

aadrekina@fedora:~/work/arch--pc/lab09 — gdb lab09-2
~/work/arch--pc/lab09
aadrekina@fedora:~/work/arch--pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aadrekina@fedora:~/work/arch--pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
aadrekina@fedora:~/work/arch--pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 16.2-3.fc42

```

Рисунок 4.2: Создание исполняемого файла с ключом „-g“.

Затем я загрузила исполняемый файл в отладчик gdb. (Рисунок 4.3)

```
aadrekina@fedora:~/work/arch--pc/lab09$ cd ~/work/arch--pc/lab09
aadrekina@fedora:~/work/arch--pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 16.2-3.fc42
Copyright (C) 2024 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 lab09-2...
(gdb) _
```

Рисунок 4.3: Загрузка исполняемого файла.

Затем, чтобы проверить работу программы, я запустила эту программу в оболочке GDB с помощью команды `run`. (рисунок 4.4)

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

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 51.96 K separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 9827) exited normally]
(gdb) █
```

Рисунок 4.4: Запуск программы в оболочке GDB.

Затем, чтобы более подробно проанализировать эту программу я установила брейк-поинт на метку `_start`. (Рисунок 4.5)

```

(gdb) break _start
Breakpoint 1 at 0x8048080: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/aadrekina/work/arch--pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)

```

Рисунок 4.5: Установка брейкпоинта.

Затем я посмотрела дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`. (Рисунок 4.6)

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:      mov     $0x4,%eax
      0x08048085 <+5>:      mov     $0x1,%ebx
      0x0804808a <+10>:     mov     $0x8049000,%ecx
      0x0804808f <+15>:     mov     $0x8,%edx
      0x08048094 <+20>:     int     $0x80
      0x08048096 <+22>:     mov     $0x4,%eax
      0x0804809b <+27>:     mov     $0x1,%ebx
      0x080480a0 <+32>:     mov     $0x8049008,%ecx
      0x080480a5 <+37>:     mov     $0x7,%edx
      0x080480aa <+42>:     int     $0x80
      0x080480ac <+44>:     mov     $0x1,%eax
      0x080480b1 <+49>:     mov     $0x0,%ebx
      0x080480b6 <+54>:     int     $0x80
End of assembler dump.
(gdb)

```

Рисунок 4.6: Просмотр дисассимилированного кода программы.

Затем я переключилась на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`. (Рисунок 4.7)

```

(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov     eax,0x4
    0x08048085 <+5>:    mov     ebx,0x1
    0x0804808a <+10>:   mov     ecx,0x8049000
    0x0804808f <+15>:   mov     edx,0x8
    0x08048094 <+20>:   int     0x80
    0x08048096 <+22>:   mov     eax,0x4
    0x0804809b <+27>:   mov     ebx,0x1
    0x080480a0 <+32>:   mov     ecx,0x8049008
    0x080480a5 <+37>:   mov     edx,0x7
    0x080480aa <+42>:   int     0x80
    0x080480ac <+44>:   mov     eax,0x1
    0x080480b1 <+49>:   mov     ebx,0x0
    0x080480b6 <+54>:   int     0x80
End of assembler dump.
(gdb)

```

Рисунок 4.7: Переключение на отображение команд с Intel'овским синтаксисом.

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel заключается в том, что в Intel синтаксические операнды идут в порядке dst,src и еще регистры пишутся без %. А в АТТ синтаксис обратный, сначала src, а потом dst и при этом используется % для регистров и \$ для констант.

Затем я включила режим псевдографики, чтобы анализировать программы было удобнее. (Рисунок 4.8)



```
~/work/arch--pc/lab09

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffce60 0xffffce60
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8048080 0x8048080 <_start>
eflags   0x202    [ IF ]
cs       0x23     35

B> 0x8048080 <_start> mov eax,0x4
0x8048085 <_start+5> mov ebx,0x1
0x804808a <_start+10> mov ecx,0x8049000
0x804808f <_start+15> mov edx,0x8
0x8048094 <_start+20> int 0x80
0x8048096 <_start+22> mov eax,0x4
0x804809b <_start+27> mov ebx,0x1
0x80480a0 <_start+32> mov ecx,0x8049008
0x80480a5 <_start+37> mov edx,0x7
0x80480aa <_start+42> int 0x80
0x80480ac <_start+44> mov eax,0x1
0x80480b1 <_start+49> mov ebx,0x0

native process 10174 (asm) In: _start L9 PC: 0x8048080
(gdb) layout regs
(gdb)
```

Рисунок 4.8: Включение режима псевдографики.

## 5 Добавление точек останова.

С помощью команды `info breakpoints` (кратко `i b`) я проверила установлена ли точка останова. (Рисунок 5.1)

```
(gdb) layout regs
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y   0x08048080 lab09-2.asm:9
breakpoint already hit 1 time
(gdb)
```

Рисунок 5.1: Проверка точки останова.

Затем я установила еще одну точку останова по адресу инструкции. Адрес я взяла у предпоследней инструкции (`mov ebx, 0x0`) (Рисунок 5.2)

```
(gdb) layout regs
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y   0x08048080 lab09-2.asm:9
breakpoint already hit 1 time
(gdb) b *<0x080480b1>
A syntax error in expression, near `<0x080480b1>'.
(gdb) b *0x080480b1
Breakpoint 2 at 0x080480b1: file lab09-2.asm, line 20.
(gdb)
```

Рисунок 5.2: Установление еще одной точки останова.

Затем я еще раз ввела команду `info breakpoints`, чтобы посмотреть информацию о всех установленных точках останова. (Рисунок 5.3)

```

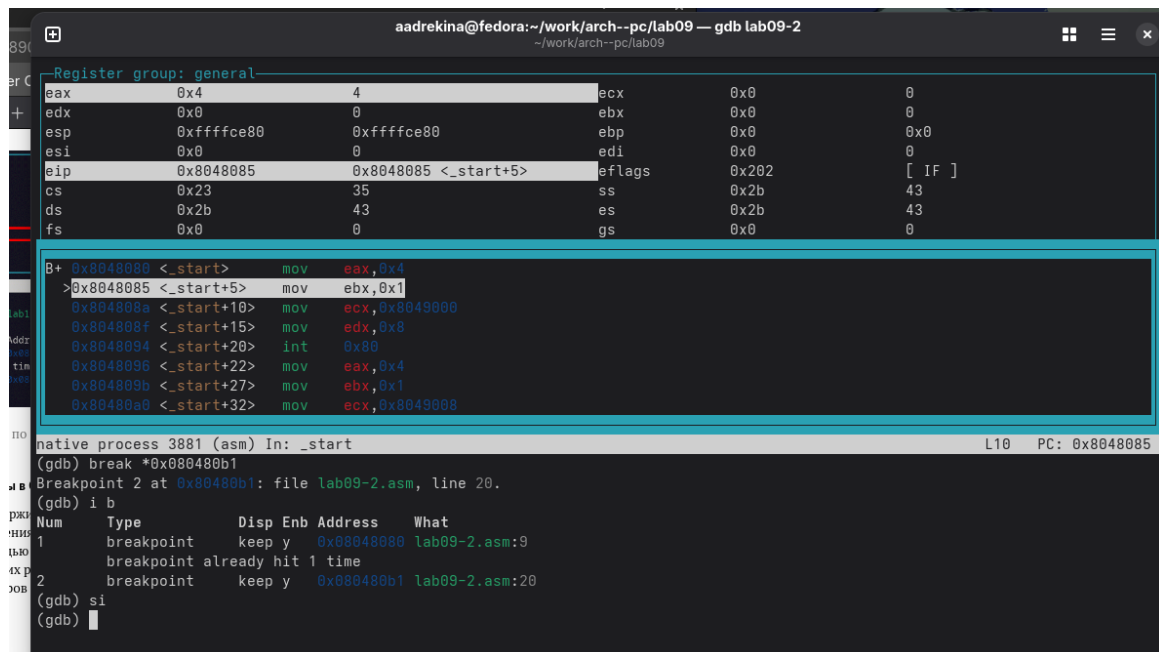
breakpoint already hit 1 time
(gdb) b *<0x080480b1>
A syntax error in expression, near `<0x080480b1>'.
(gdb) b *0x080480b1
Breakpoint 2 at 0x080480b1: file lab09-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y  0x08048080 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint      keep y  0x080480b1 lab09-2.asm:20
(gdb) █

```

Рисунок 5.3: Проверка точек останова.

## 6 Работа с данными программы в GDB.

Я выполнила 5 инструкций с помощью команды step(или si). (Рисунок 6.1), (Рисунок 6.2),(Рисунок 6.3),(Рисунок 6.4),(Рисунок 6.5)



The screenshot shows the GDB interface with the following content:

```
aadrekina@fedora:~/work/arch--pc/lab09 — gdb lab09-2
~/work/arch--pc/lab09

Register group: general
eax 0x4 4 ecx 0x0 0
edx 0x0 0 ebx 0x0 0
esp 0xffffce80 0xffffce80 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x8048085 0x8048085 <_start+5> eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

0x8048080 <_start> mov eax,0x4
>0x8048085 <_start+5> mov ebx,0x1
0x804808a <_start+10> mov ecx,0x8049000
0x804808f <_start+15> mov edx,0x8
0x8048094 <_start+20> int 0x80
0x8048096 <_start+22> mov eax,0x4
0x804809b <_start+27> mov ebx,0x1
0x80480a0 <_start+32> mov ecx,0x8049008

native process 3881 (asm) In: _start L10 PC: 0x8048085
(gdb) break *0x80480b1
Breakpoint 2 at 0x80480b1: file lab09-2.asm, line 20.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8048080 lab09-2.asm:9
breakpoint already hit 1 time
2 breakpoint keep y 0x80480b1 lab09-2.asm:20
(gdb) si
(gdb) 
```

Рисунок 6.1: Первая инструкция.

native process 3881 (asm) In: \_start L11 PC: 0x804808a

```
(gdb) break *0x080480b1
Breakpoint 2 at 0x080480b1: file lab09-2.asm, line 20.
(gdb) i b
Num    Type      Disp Enb Address  What
1      breakpoint keep y  0x08048080 lab09-2.asm:9
       breakpoint already hit 1 time
2      breakpoint keep y  0x080480b1 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb)
```

Рисунок 6.2: Вторая инструкция.

native process 3881 (asm) In: \_start L12 PC: 0x804808f

```
(gdb) i b
Num    Type      Disp Enb Address  What
1      breakpoint keep y  0x08048080 lab09-2.asm:9
       breakpoint already hit 1 time
2      breakpoint keep y  0x080480b1 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb)
```

Рисунок 6.3: Третья инструкция.

```

aadrekina@fedora:~/work/arch--pc/lab09 — gdb lab09-2
~/work/arch--pc/lab09

Register group: general
eax      0x4      4      ecx      0x8049000      134516736
edx      0x8      8      ebx      0x1      1
esp      0xffffce80      0xffffce80      ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8048094      0x8048094 <_start+20>      eflags      0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       0x0      0

0+ 0x8048080 <_start>      mov     eax,0x4
0x8048085 <_start+5>      mov     ebx,0x1
0x804808a <_start+10>     mov     ecx,0x8049000
0x804808f <_start+15>     mov     edx,0x8
>0x8048094 <_start+20>     int     0x80
0x8048096 <_start+22>     mov     eax,0x4
0x804809b <_start+27>     mov     ebx,0x1
0x80480a0 <_start+32>     mov     ecx,0x8049008

native process 3881 (asm) In: _start L13 PC: 0x8048094
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y 0x08048080 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint keep y 0x080480b1 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рисунок 6.4: Четвертая инструкция.

```

aadrekina@fedora:~/work/arch--pc/lab09 — gdb lab09-2
~/work/arch--pc/lab09

Register group: general
eax      0x8      8      ecx      0x8049000      134516736
edx      0x8      8      ebx      0x1      1
esp      0xffffce80      0xffffce80      ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8048096      0x8048096 <_start+22>      eflags      0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       0x0      0

0+ 0x8048080 <_start>      mov     eax,0x4
0x8048085 <_start+5>      mov     ebx,0x1
0x804808a <_start+10>     mov     ecx,0x8049000
0x804808f <_start+15>     mov     edx,0x8
0x8048094 <_start+20>     int     0x80
>0x8048096 <_start+22>     mov     eax,0x4
0x804809b <_start+27>     mov     ebx,0x1
0x80480a0 <_start+32>     mov     ecx,0x8049008

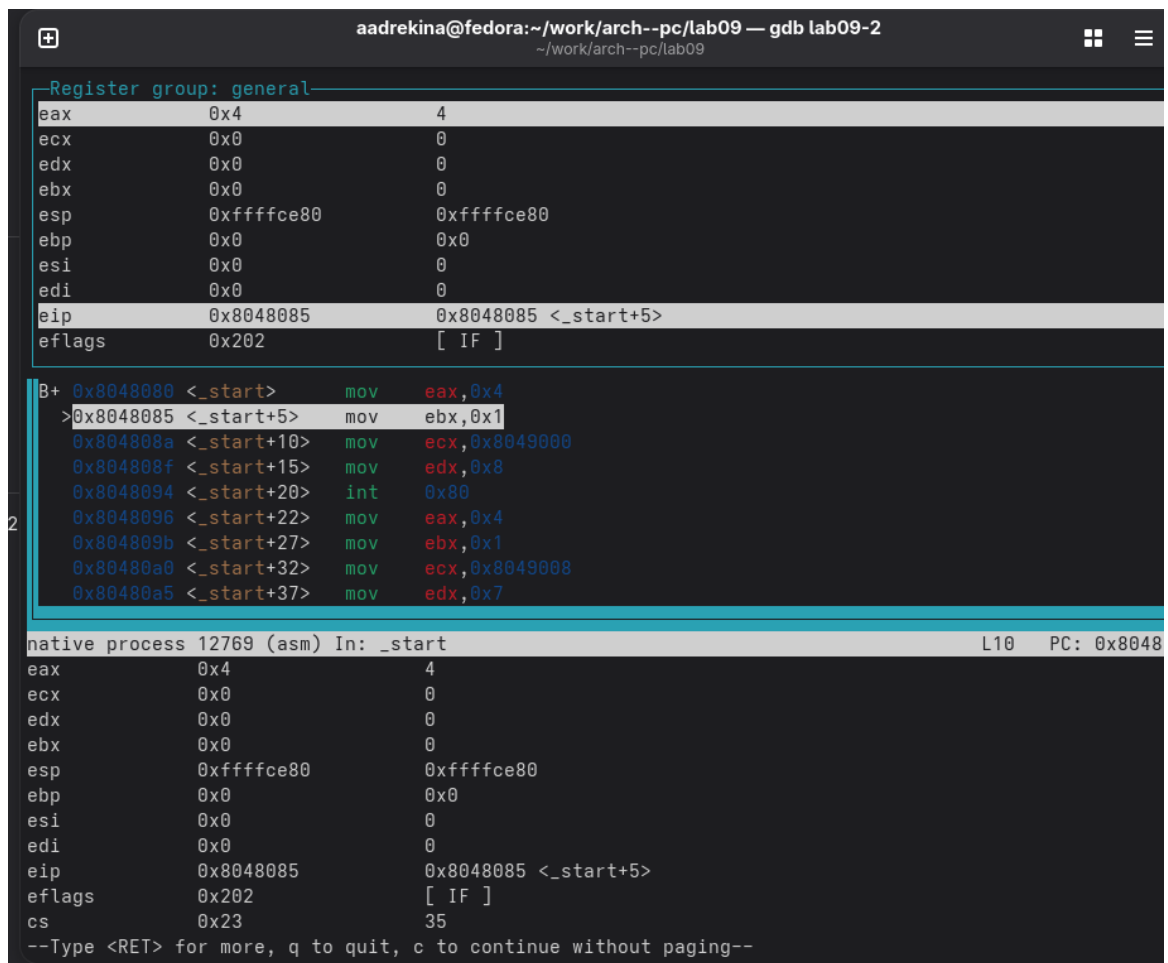
native process 3881 (asm) In: _start L14 PC: 0x8048096
Num      Type      Disp Enb Address      What
1        breakpoint keep y 0x08048080 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint keep y 0x080480b1 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) siHello,
(gdb)

```

Рисунок 6.5: Пятая инструкция.

Я внимательно изучила эти 5 скриншоты, и проследила за изменением регистров. Меняются такие регистры как: EAX,EBX,ECX,EDX.

Я посмотрела содержимое регистров с помощью команды `info registers`. (Рисунок 6.6)



The screenshot shows a GDB terminal window with the title `aadrekina@fedora:~/work/arch--pc/lab09 — gdb lab09-2`. The window displays the state of general-purpose registers and the current assembly instructions.

**Register group: general**

| Register | Value      | Comment              |
|----------|------------|----------------------|
| eax      | 0x4        | 4                    |
| ecx      | 0x0        | 0                    |
| edx      | 0x0        | 0                    |
| ebx      | 0x0        | 0                    |
| esp      | 0xffffce80 | 0xffffce80           |
| ebp      | 0x0        | 0x0                  |
| esi      | 0x0        | 0                    |
| edi      | 0x0        | 0                    |
| eip      | 0x8048085  | 0x8048085 <_start+5> |
| eflags   | 0x202      | [ IF ]               |

**Assembly instructions:**

```
B+ 0x8048080 <_start>    mov     eax,0x4
>0x8048085 <_start+5>    mov     ebx,0x1
0x804808a <_start+10>    mov     ecx,0x8049000
0x804808f <_start+15>    mov     edx,0x8
0x8048094 <_start+20>    int     0x80
0x8048096 <_start+22>    mov     eax,0x4
0x804809b <_start+27>    mov     ebx,0x1
0x80480a0 <_start+32>    mov     ecx,0x8049008
0x80480a5 <_start+37>    mov     edx,0x7
```

**Native process 12769 (asm) In: \_start**

| Register | Value      | Comment              |
|----------|------------|----------------------|
| eax      | 0x4        | 4                    |
| ecx      | 0x0        | 0                    |
| edx      | 0x0        | 0                    |
| ebx      | 0x0        | 0                    |
| esp      | 0xffffce80 | 0xffffce80           |
| ebp      | 0x0        | 0x0                  |
| esi      | 0x0        | 0                    |
| edi      | 0x0        | 0                    |
| eip      | 0x8048085  | 0x8048085 <_start+5> |
| eflags   | 0x202      | [ IF ]               |
| cs       | 0x23       | 35                   |

--Type <RET> for more, q to quit, c to continue without paging--

Рисунок 6.6: Просмотр содержимого регистра.

Потом я посмотрела значение переменной `msg1` по имени. (Рисунок 6.7)

```
aadrekina@fedora:~/work/arch--pc/lab09 — gdb lab09-2
~/work/arch--pc/lab09

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x1      1
esp      0xffffce80 0xffffce80
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x804808a 0x804808a <_start+10>
eflags   0x202    [ IF ]

B+ 0x8048080 <_start>    mov     eax,0x4
0x8048085 <_start+5>    mov     ebx,0x1
>0x804808a <_start+10>   mov     ecx,0x8049000
0x804808f <_start+15>   mov     edx,0x8
0x8048094 <_start+20>   int     0x80
0x8048096 <_start+22>   mov     eax,0x4
0x804809b <_start+27>   mov     ebx,0x1
0x80480a0 <_start+32>   mov     ecx,0x8049008
0x80480a5 <_start+37>   mov     edx,0x7

native process 13524 (asm) In: _start L11 PC: 0x80480
eip      0x804808a 0x804808a <_start+10>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) x/1sb &msg1
0x8049000 <msg1>: "Hello, "
(gdb)
```

Рисунок 6.7: Значение переменной msg1.

У меня вывелся такой же ответ, как и в лекции, значит я сделала все правильно.

Потом, таким же образом я посмотрела значение переменной msg2 по адресу. (Рисунок 6.8)

```
(gdb) x/1sb 0x8049008
0x8049008 <msg2>: "world!\n\034"
(gdb)
```

Рисунок 6.8: Значение переменной msg2.

Затем я изменила значение для регистра с помощью команды set. (Рисунок 6.9)



```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x8049000 <msg1>: "hello, "
```

Рисунок 6.9: Изменение значений для регистра msg1.

Ответ вывелся такой же, значит я сделал все правильно.

Затем я также изменила значение для регистра msg2. (Рисунок 6.10)

```
(gdb) set {char}&msg2='W'
(gdb) x/1sb &msg2
0x8049008 <msg2>: "World!\n\034"
```

Рисунок 6.10: Изменения значений для регистра msg2.

Затем с помощью команды set я изменила значение регистра. (Рисунок 6.11)

```
native process 13524 (asm) In: _start L11 PC: 0x80480
(gdb) set $ebx=2
(gdb) p/s $ebx
$2 = 2
(gdb) p/s $eax
$3 = 4
(gdb) p/t $eax
$4 = 100
(gdb) p/s $ecx
$5 = 0
```

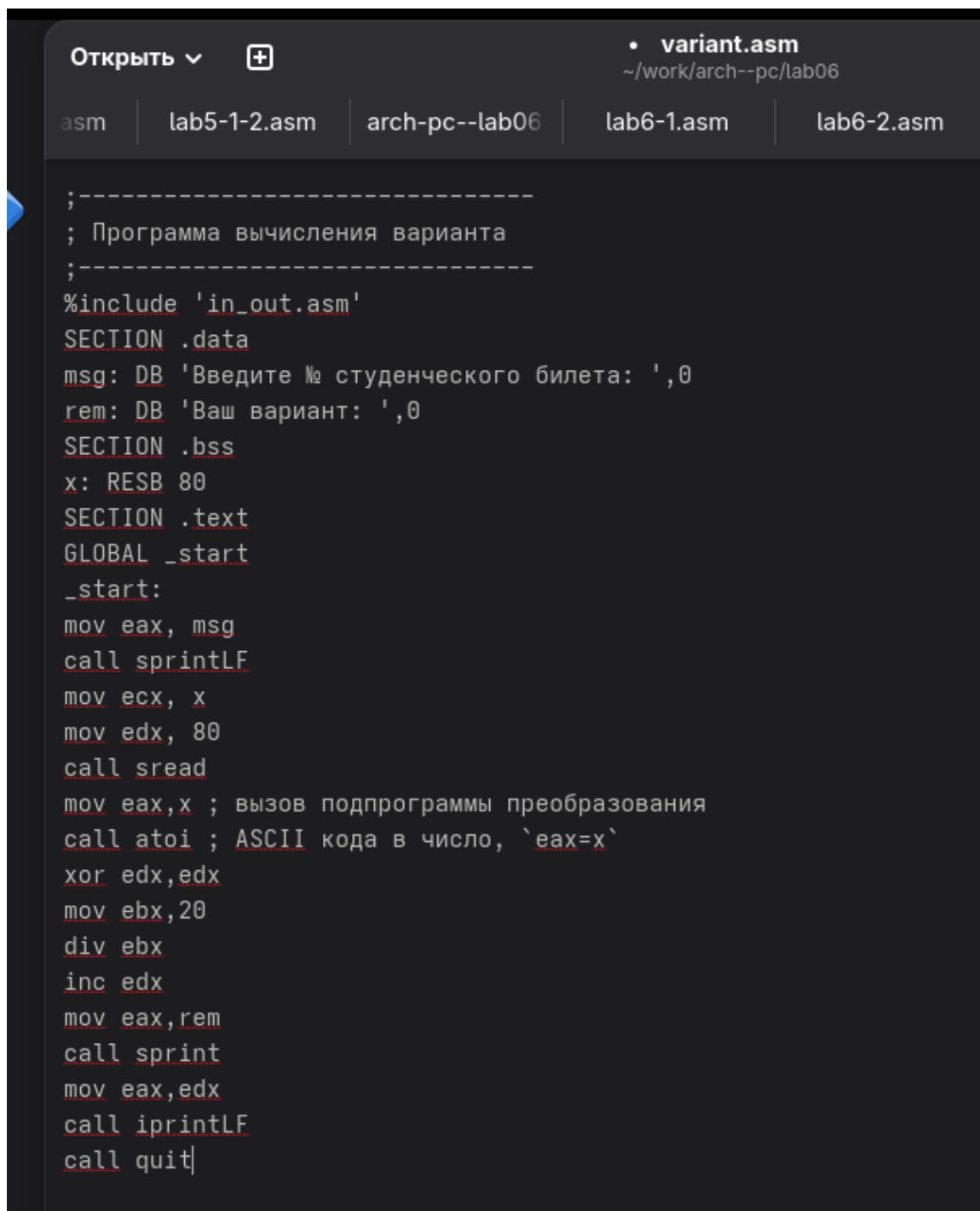
Рисунок 6.11: Изменение значение регистра.

Команда p/s \$ebx выводит строку по адресу, хранящемуся в ebx, следовательно при set \$ebx=«2» показывается ASCLL-код символа, а при set \$ebx=2 - числовое значение. Команда p/t \$ebx выводит значение регистра в двоичном коде.

В конце я завершила выполнение программы с помощью команды continue и вышла из GDB с помощью команды quit.

## **7 Обработка аргументов командной строки в GDB.**

Я скопировала файл lab8-2.asm, который был создан при выполнении лабораторной работы №8 в файл lab09-3.asm. Затем я создала исполняемый файл (Рисунок 7.1)




```
Открыть ▾  • variant.asm  
~/work/arch--pc/lab06  
asm | lab5-1-2.asm | arch-pc--lab06 | lab6-1.asm | lab6-2.asm  
;-----  
; Программа вычисления варианта  
;-----  
%include 'in_out.asm'  
SECTION .data  
msg: DB 'Введите № студенческого билета: ',0  
rem: DB 'Ваш вариант: ',0  
SECTION .bss  
x: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, msg  
call sprintf  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, `eax=x`  
xor edx, edx  
mov ebx, 20  
div ebx  
inc edx  
mov eax, rem  
call sprintf  
mov eax, edx  
call iprintf  
call quit
```

Рисунок 7.1: Копирование файла.

Чтобы загрузить в GDB программы с аргументами необходимо использовать ключ `-args` (Рисунок 7.2)

```
aadrekina@fedora:~/work/arch--pc/lab09$ cd ~/lab09-3
aadrekina@fedora:~/work/arch--pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 16.2-3.fc42
Copyright (C) 2024 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...
```

Рисунок 7.2: Загрузка исполняемого файла в отладчик, указав аргументы.

Потом я установила точку останова перед первой инструкцией и запустила ее. (Рисунок 7.3)

```
(gdb) b _start
Breakpoint 1 at 0x8048148: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/aadrekina/work/arch--pc/lab09/lab09-3 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.

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

Рисунок 7.3: Установка точки останова и запуск.

Я посмотрела позиции стека - по адресу [esp+4].(Рисунок 7.4)

```

debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xfffffce40: 0x00000005
(gdb) x/s *(void**)( $esp + 4)
0xfffffd010: "/home/aadrekina/work/arch--pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8)
0xfffffd03c: "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xfffffd04e: "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xfffffd05f: "2"
(gdb) x/s *(void**)( $esp + 20)
0xfffffd061: "аргумент 3"
(gdb) x/s *(void**)( $esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рисунок 7.4: Просмотр остальных позиций стека.

Шаг изменения равен 4, так как в данной программе каждый адрес в памяти занимает 4 байта, стек хранит обычный список адресов, и для того чтобы перейти к следующему адресу нужно сдвинуться ровно на 4 байта.

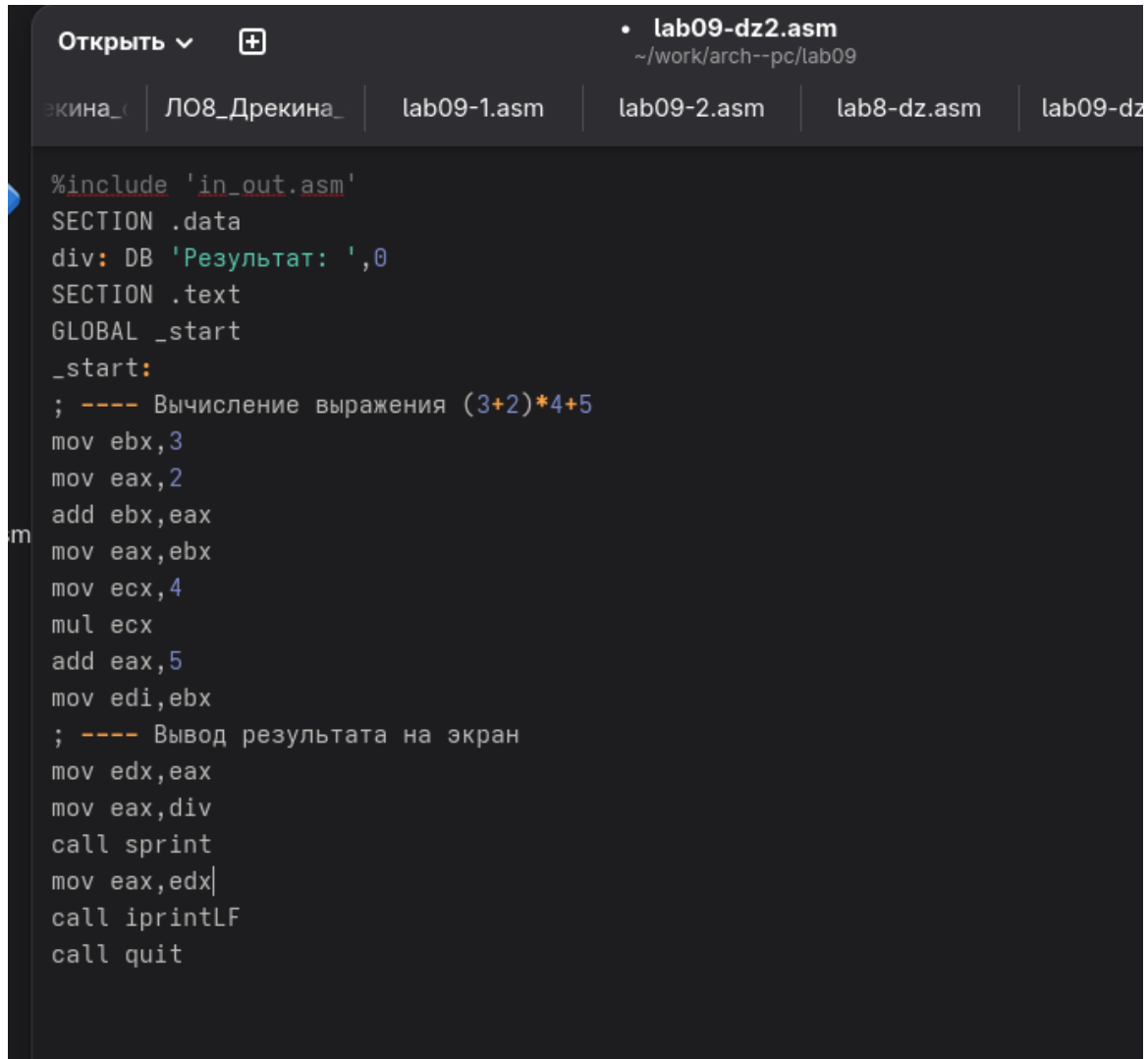
## **8 Задание для самостоятельной работы.**

Я преобразовала программу созданную во время выполнения лабораторной работы №8, реализовав вычисления функции  $f(x)$  как подпрограмму. (Рисунок 8.1)



Рисунок 8.1: Преобразования программы.

Я создала еще один текстовый файл и вставила туда Листинг 9.3. Я начала анализировать эту программу с помощью отладчика GDB, определила ошибки и исправила их. (Рисунок 8.2)




```
Открыть ▾  • lab09-dz2.asm  
~/work/arch--pc/lab09  
экина_ | ЛО8_Дрекина_ | lab09-1.asm | lab09-2.asm | lab8-dz.asm | lab09-dz2.asm  
  
%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 eax,ebx  
mov ecx,4  
mul ecx  
add eax,5  
mov edi,ebx  
; ---- Вывод результата на экран  
mov edx,eax  
mov eax,div  
call sprint  
mov eax,edx  
call iprintLF  
call quit
```

Рисунок 8.2: Изменение в программе.

Листинг 9.3:

```
%include 'in_out.asm'  
SECTION .data  
div: DB 'Результат: ',0
```

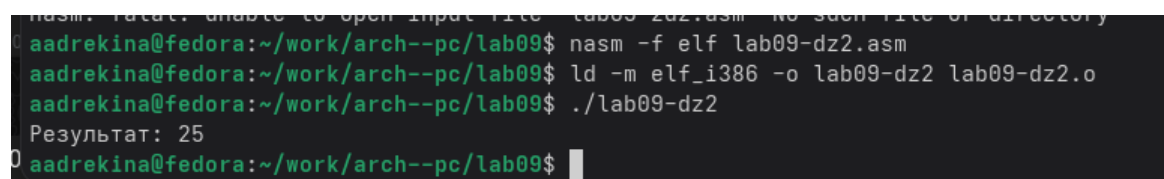


```

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

```

Затем я сделала файл исполняемым и проверила результат, ответы совпали. Значит я правильно изменила программу.



```

nasm: fatal: unable to open input file 'lab09-dz2.asm': no such file or directory
0 aadrekina@fedora:~/work/arch--pc/lab09$ nasm -f elf lab09-dz2.asm
aadrekina@fedora:~/work/arch--pc/lab09$ ld -m elf_i386 -o lab09-dz2 lab09-dz2.o
aadrekina@fedora:~/work/arch--pc/lab09$ ./lab09-dz2
Результат: 25
0 aadrekina@fedora:~/work/arch--pc/lab09$

```

Рисунок 8.3: Запуск.

## 9 Вывод.

Я приобрела навыки написания программы с использованием подпрограммы. И ознакомилась с методами отладки при помощи GDB и его основными возможностями.