

# **Лабораторная работа №9**

Воинов Кирилл Викторович

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы	18
4	Выводы	26

# Список иллюстраций

2.1	Создание каталога, переход в него и создание файла . . . . .	5
2.2	Текст программы . . . . .	6
2.3	Работа программы . . . . .	7
2.4	Текст измененной программы . . . . .	8
2.5	Создание файла . . . . .	9
2.6	Текст программы . . . . .	10
2.7	Получение исполняемого файла и загрузка его в отладчик gdb. . .	11
2.8	Проверка работы программы . . . . .	11
2.9	Установка брейкпоинта и запуск программы . . . . .	11
2.10	Дисассимилированный код программы . . . . .	12
2.11	Отображение команд с Intel'овским синтаксисом . . . . .	13
2.12	Режим псевдографики . . . . .	14
2.13	Проверяю наличие брейкпоинта и установка нового . . . . .	14
2.14	Информация о брейкпоинтах . . . . .	14
2.15	Значение переменной msg1 . . . . .	15
2.16	Изменение первого символа переменной msg1 . . . . .	15
2.17	Изменение первого символа переменной msg2 . . . . .	15
2.18	Изменение значения регистра ebx . . . . .	16
2.19	Копия файла lab8-2.asm . . . . .	16
2.20	Создание исполняемого файла . . . . .	16
2.21	Загрузка исполняемого файла . . . . .	17
2.22	Точка остановки . . . . .	17
2.23	Запуск программы и просмотр элементов стека. . . . .	17
3.1	Текст программы . . . . .	19
3.2	Работа программы . . . . .	20
3.3	Текст программы . . . . .	21
3.4	До умножения . . . . .	22
3.5	После умножения . . . . .	23
3.6	Текст измененной программы . . . . .	24
3.7	Работа измененной программы . . . . .	25

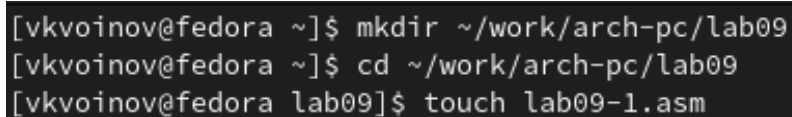
# 1 Цель работы

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

## 2 Выполнение лабораторной работы

### 1. Реализация подпрограмм в NASM

- 1) Создаю каталог для выполнения лабораторной работы No 9, перехожу в него и создаю файл lab09-1.asm.(рис. 2.1).



```
[vkvoinov@fedora ~]$ mkdir ~/work/arch-pc/lab09  
[vkvoinov@fedora ~]$ cd ~/work/arch-pc/lab09  
[vkvoinov@fedora lab09]$ touch lab09-1.asm
```

Рис. 2.1: Создание каталога, переход в него и создание файла

- 2) Ввожу в файл lab09-1.asm текст программы из листинга 9.1.(рис. 2.2).

```

%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. 2.2: Текст программы

Создаю исполняемый файл и проверьте его работу.(рис. 2.3).

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

Рис. 2.3: Работа программы

Изменяю текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ . Т.е.  $x$  передается в подпрограмму `_calcul` из нее в подпрограмму `_subcalcul`, где вычисляется выражение  $g(x)$ , результат возвращается в `_calcul` и вычисляется выражение  $f(g(x))$ . Результат возвращается в основную программу для вывода результата на экран.(рис. 2.4) и (рис. 2.4).

```

%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
subres: 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:
call _subcalcul
mov eax, [subres]
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
;-----
; Подпрограмма для подпрограммы вычисления
; выражения "3x-1"
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
mov [sub|res], eax
ret ; выход из подпрограммы

```

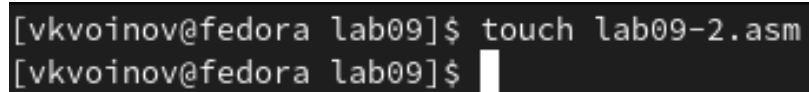
Рис. 2.4: Текст измененной программы



[Работа измененной программы]](image/5.png){#fig:005 width=70%}

## 2. Отладка программ с помощью GDB

- 1) Создаю файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!).(рис. 2.5) и (рис. 2.6).



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

Рис. 2.5: Создание файла

```

SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1Len
int 0x80
mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2Len
int 0x80
mov eax, 1
mov ebx, 0
int 0x80

```

Рис. 2.6: Текст программы

Получаю исполняемый файл и загружаю его в отладчик gdb.(рис. 2.7).

```
[vkvoinov@fedora lab09]$ nasm -f elf lab09-2.asm
[vkvoinov@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o
[vkvoinov@fedora lab09]$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
[vkvoinov@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o
[vkvoinov@fedora lab09]$ gdb lab09-2
GNU gdb (GDB) Fedora Linux 13.1-2.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 lab09-2...
(gdb)
```

Рис. 2.7: Получение исполняемого файла и загрузка его в отладчик gdb.

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run.(рис. 2.8).

```
[Inferior 1 (process 4649) exited normally]
(gdb) run
Starting program: /home/vkvoinov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4700) exited normally]
(gdb)
```

Рис. 2.8: Проверка работы программы

Устанавливаю брейкпоинт на метку \_start и запускаю программу.(рис. 2.9).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/vkvoinov/work/arch-pc/lab09/lab09-2

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

Рис. 2.9: Установка брейкпоинта и запуск программы

С помощью команды disassemble начиная с метки \_start смотрю дисассимилированный код программы.(рис. 2.10).

```

(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. 2.10: Дисассимилированный код программы

Переключаюсь на отображение команд с Intel'овским синтаксисом.(рис. 2.11).

```

(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. 2.11: Отображение команд с Intel'овским синтаксисом

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel: Различия присутствуют лишь конце строчек. Перед операндами и адресами в первом случае ставятся % и \$ соответственно и первым стоит адрес, а во втором таких символов нет и первым стоит адрес. (например \$0x4,%eax и eax,0x4)

Включаю режим псевдографики для более удобного анализа программы. (рис. 2.12).

```

[ Register Values Unavailable ]

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 4728 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb)

```

Рис. 2.12: Режим псевдографики

2)Проверяю наличие брейкпоинта по имени метки (\_start) и устанавливаю новую по адресу предпоследней инструкции.(рис. 2.13).

```

(gdb) layout regs
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb)

```

Рис. 2.13: Проверка наличия брейкпоинта и установка нового

Просматриваю информацию о всех установленных брейкпоинтах.(рис. 2.14).

```

(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. 2.14: Информация о брейкпоинтах

При выполнении 5 инструкций изменяются регистры eax, ebx, ecx, edx, eax. Смотрю значение переменной msg1.(рис. 2.15).

```
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) █
```

Рис. 2.15: Значение переменной msg1

Изменяю первый символ переменной msg1.(рис. 2.16).

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) █
```

Рис. 2.16: Изменение первого символа переменной msg1

Изменяю первый символ переменной msg2.(рис. 2.17).

```
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg2='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "World!\n\034"
(gdb) █
```

Рис. 2.17: Изменение первого символа переменной msg2

С помощью команды set изменяю значение регистра ebx.(рис. 2.18).

```

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

```

Рис. 2.18: Изменение значения регистра ebx

Разница возникает из за того, что в первом случае 2 читается как символ а во втором как число.

Завершаю выполнение программы и выхожу из GDB.

- 3) Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы No8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm. (рис. 2.19).

```

[vkvoinov@fedora lab09]$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
[vkvoinov@fedora lab09]$

```

Рис. 2.19: Копия файла lab8-2.asm

Создаю исполняемый файл.(рис. 2.20).

```

[vkvoinov@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
[vkvoinov@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o

```

Рис. 2.20: Создание исполняемого файла

Загружаю исполняемый файл в отладчик, указав аргументы. (рис. 2.21).



```
[vkvoinov@fedora lab09]$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora Linux 13.1-2.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. 2.21: Загрузка исполняемого файла

Устанавливаю точку останова перед первой инструкцией в программе.(рис. 2.22).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
```

Рис. 2.22: Точка останова

Запускаю программу и просматриваю позиции стека.(рис. 2.23).

```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/vkvoinov/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffd1a0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd340: "/home/vkvoinov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd370: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd385: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd390: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd390: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.23: Запуск программы и просмотр элементов стека.

### **3 Задание для самостоятельной работы**

1.Преобразовываю программу из лабораторной работы No8 (Задание No1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму.(рис. 3.1) и (рис. 3.2).

```

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msv db "f(x)=17+5x",0
SECTION .text
global _start
mov eax, msv
call iprintLF
_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`)
mov ebx,5
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mul ebx
add eax, 17
add esi, eax
; след. аргумент `esi=esi+f`
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

```

Рис. 3.1: Текст программы

```
[vkvoinov@fedora lab09]$ nasm -f elf lab8-4-18.asm
[vkvoinov@fedora lab09]$ ld -m elf_i386 -o lab8-4-18 lab8-4-18.o
[vkvoinov@fedora lab09]$ ./lab8-4-18 1
Результат: 22
[vkvoinov@fedora lab09]$ ./lab8-4-18 5
Результат: 42
[vkvoinov@fedora lab09]$ ./lab8-4-18 1 2 3
Результат: 81
[vkvoinov@fedora lab09]$
```

Рис. 3.2: Работа программы

В листинге 9.3 приведена программа вычисления выражения  $(3+2)^*4+5$ . При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.(рис. 3.4).

```

%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. 3.3: Текст программы

С помощью отладчика я поочередно проверял каждую строку и изменение регистров и заметил, что при умножении на ecx, ebx не изменяется, а изменяется eax. (рис. 3.4) и (рис. 3.5).

eax	0x8	8	ecx	0x4	4
eax	0x2	2	ecx	0x4	4
edx	0xffffd1c0	0xffffd1c0<_start>	ebx	0x5	5
esp	0xffffd1c0	0xffffd1c0<_start+24>	ebp	0x0	0x0
esi	0x0049100	0x0049100<_start+24>	edi	0x0	0
eip	0x00490f9	0x00490f9<_start+17>	eflags	0x206	[ PF IF ]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	0x0	0

```

b- 0x00490e8<_start> mov ebx,0x3

B+ 0x00490e8<_start> mov ebx,0x3
0x00490ed<_start+5> mov eax,0x2
0x00490f2<_start+10> add ebx,eax
B+ 0x00490f2<_start+12> mov ecx,0x4
> 0x00490f9<_start+17> mul ecx
0x00490fb<_start+19> add eax,0x5
b+ 0x00490fe<_start+22> mov edi,ebx
b+ 0x0049100<_start+24> mov eax,0x0049000
0x0049105<_start+29> call 0x004900f<sprint>
0x004910a<_start+34> mov eax,edi
0x004910c<_start+36> call 0x0049086<iprintLF>
0x0049111<_start+41> call 0x00490db<quit>
0x0049116 add BYTE PTR [eax],al
0x0049118 add BYTE PTR [eax],al
0x004911a add BYTE PTR [eax],al
0x004911c add BYTE PTR [eax],al

```

L16 PC: 0x0049100  
L12 PC: 0x00490f9

```

process 5532 In: _start
Breakpoint process 5551 In: _start@9-4.asm:8
Breakpoint 2, _start () at lab09-4.asm:16
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 5532) exited normally]
(gdb) run
Starting program: /home/vkvoinov/work/arch-pc/lab09/lab09-4

Breakpoint 1, _start () at lab09-4.asm:8
(gdb) c
Continuing.

Breakpoint 3, _start () at lab09-4.asm:11
(gdb) sI
Undefined command: "sI". Try "help".
(gdb) stepi
(gdb)

```

Рис. 3.4: До умножения

```

eax    0x8      8      ecx    0x4      4
eax    0x8      8      ecx    0x4      4
edx    0xffffd1c0 0xffffd1c0<_start> ebx    0x5      5
esp    0xffffd1c0 0xffffd1c0<_start+24> ebp    0x0      0x0
esi    0x0049100 0x0049100<_start+24> edi    0x0      0
ebp    0x00490fb 0x00490fb<_start+19> eflags 0x202    [ IF ]
cs     0x23     35     ss     0x2b     43
ds     0x2b     43     es     0x2b     43
fs     0x0      0      gs     0x0      0

b- 0x00490eb<_start> mov ebx,0x3

B+ 0x00490eb<_start> mov ebx,0x3
0x00490ed<_start+5> mov eax,0x2
0x00490f2<_start+10> add ebx,eax
B+ 0x00490f4<_start+12> mov ecx,0x4
0x00490f9<_start+17> mul ecx
> 0x00490fb<_start+19> add ebx,0x5
b+ 0x00490fe<_start+22> mov edi,ebx
b+ 0x0049100<_start+24> mov eax,0x0049000
0x0049105<_start+29> call 0x004900f<sprint>
0x004910a<_start+34> mov eax,edi
0x004910c<_start+36> call 0x0049006<iprintLF>
0x0049111<_start+41> call 0x004900b<quit>
0x0049116 add BYTE PTR [eax],al
0x0049118 add BYTE PTR [eax],al
0x004911a add BYTE PTR [eax],al
0x004911c add BYTE PTR [eax],al

process 5532 In: _start
Breakpoint process 5551 In: _start@9-4.asm:8
(gdb) c
Continuing.
Pezyun'at: 10
[Inferior 1 (process 5532) exited normally]
(gdb) run
Starting program: /home/vkvoinov/work/arch-pc/lab09/lab09-4

Breakpoint 1, _start () at lab09-4.asm:8
(gdb) c
Continuing.

Breakpoint 3, _start () at lab09-4.asm:11
(gdb) sI
Undefined command: "sI". Try "help".
(gdb) stepi
(gdb) stepi
(gdb)

```

Рис. 3.5: После умножения

Для исправления этой ошибки достаточно поменять местами ebx и eax.(рис. 3.6) и (рис. 3.7).

```
%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 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 iprintfLF
call quit
```

Рис. 3.6: Текст измененной программы

(рис. 3.7).



```
[vkvoinov@fedora lab09]$ nasm -f elf lab09-4.asm  
[vkvoinov@fedora lab09]$ ld -m elf_i386 -o lab09-4 lab09-4.o  
[vkvoinov@fedora lab09]$ ./lab09-4  
Результат: 25  
[vkvoinov@fedora lab09]$
```

Рис. 3.7: Работа измененной программы

## 4 Выводы

На этой лабораторной работе я приобрел навыки написания программ с использованием подпрограмм, ознакомился с методами отладки при помощи GDB и его основными возможностями.