

# **Отчёт по лабораторной работе 9**

**Дисциплина: Архитектура компьютера**

Горобцова Арина Романовна НММбд-01-24

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выводы</b>	<b>22</b>

# Список иллюстраций

2.1	Программа lab9-1.asm . . . . .	7
2.2	Запуск программы lab9-1.asm . . . . .	7
2.3	Программа lab9-1.asm . . . . .	8
2.4	Запуск программы lab9-1.asm . . . . .	8
2.5	Программа lab9-2.asm . . . . .	9
2.6	Запуск программы lab9-2.asm в отладчике . . . . .	10
2.7	Дизассемблированный код . . . . .	11
2.8	Дизассемблированный код в режиме интел . . . . .	12
2.9	Точка остановки . . . . .	13
2.10	Изменение регистров . . . . .	13
2.11	Изменение регистров . . . . .	14
2.12	Изменение значения переменной . . . . .	15
2.13	Вывод значения регистра . . . . .	16
2.14	Вывод значения регистра . . . . .	17
2.15	Вывод значения регистра . . . . .	18
2.16	Программа lab9-4.asm . . . . .	19
2.17	Запуск программы lab9-4.asm . . . . .	19
2.18	Код с ошибкой . . . . .	20
2.19	Код исправлен . . . . .	21

## **Список таблиц**

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Выполнение лабораторной работы

1. Создаем каталог для выполнения лабораторной работы № 9, переходим в него и создаем файл lab9-1.asm.
2. В качестве примера рассмотрим программу вычисления арифметического выражения  $f(x) = 2x + 7$  с помощью подпрограммы calcul. В данном примере  $x$  вводится с клавиатуры, а само выражение вычисляется в подпрограмме. (рис. 2.1) , (рис. 2.2)

```

lab09-1.asm      [-M--] 11 L:[ 12+10  22/ 39] *(332 / 530b) 0010 0x00A  [*][X]
_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, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov  ebx, 2
mul  ebx
add  eax, 7
mov  [rez], eax
ret ; выход из подпрограммы
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

```

Рис. 2.1: Программа lab9-1.asm

```

argorobcova@dk3n35 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 2
2x+7=11
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 3
2x+7=13
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ █

```

Рис. 2.2: Запуск программы lab9-1.asm

3. Изменяем текст программы, добавляем подпрограмму `subcalcul` в подпрограмму `calcul`, для вычисления выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ . (рис. 2.3), (рис. 2.4)

```
lab09-1.asm [----] 11 L:[ 12+10 22/ 39] *(332 / 530b) 0010 0x00A [*][X]
_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, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 2.3: Программа lab9-1.asm

```
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 2
2(3x-1)+7=17
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 3
2(3x-1)+7=23
argorobcova@dk3n35 ~/work/arch-pc/lab09 $
```

Рис. 2.4: Запуск программы lab9-1.asm

4. Создаем файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!). (рис. 2.5)



```
lab09-2.asm      [----]  8 L:[ 1+20  21/ 21] *(293 / 293b) <EOF>      [*][X]
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 2.5: Программа lab9-2.asm

Получаем исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’.

Загружаем исполняемый файл в отладчик gdb. Проверяем работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r). (рис. 2.6)

```
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/argorobcova/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4221) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
```

---

Рис. 2.6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы установите брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запускаем её. Посмотрим дизассемблированный код программы. (рис. 2.7), (рис. 2.8)

```

Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/argorobcova/work/arch-pc/lab0
9/lab09-2
Hello, world!
[Inferior 1 (process 4221) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/argorobcova/work/arch-pc/lab0
9/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
warning: Source file is more recent than executable.
9      mov eax, 4
(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

```

---

Рис. 2.7: Дизассемблированный код

```

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) 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. 2.8: Дизассемблированный код в режиме интел

На предыдущих шагах была установлена точка остановки по имени метки (\_start). Проверяем это с помощью команды `info breakpoints` (кратко `i b`). Устанавливаем еще одну точку остановки по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определяем адрес предпоследней инструкции (`mov ebx,0x0`) и устанавливаем точку. (рис. 2.9)

```

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 4800 In: _start L9 PC: 0x8049000
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y  0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint      keep y  0x08049031 lab09-2.asm:20
(gdb)

```

Рис. 2.9: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполняем 5 инструкций с помощью команды `stepi` (или `si`) и смотрим за изменением значений регистров. (рис. 2.10), (рис. 2.11)

```

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 4800 In: _start L9 PC: 0x8049000
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y  0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint      keep y  0x08049031 lab09-2.asm:20
(gdb) si

```

Рис. 2.10: Изменение регистров

```
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffc4e0 0xffffc4e0
ebp      0x0      0x0

0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
>0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008

native process 4800 In: _start L14 PC: 0x8049016
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab09-2.asm:20
(gdb) si
(gdb) stepi
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) █
```

Рис. 2.11: Изменение регистров

Посмотрели значение переменной msg1 по имени. Посмотрели значение переменной msg2 по адресу.

Изменили значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Изменили первый символ переменной msg1.(рис. 2.12)

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffc4e0 0xffffc4e0
ebp      0x0      0x0

0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
>0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008

native process 4800 In: _start L14 PC: 0x8049016
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/lsb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/lsb 0x804a008
0x804a008 <msg2>: "Lorld!\n\034"
(gdb)

```

Рис. 2.12: Изменение значения переменной

Вывели в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx. (рис. 2.13)

```
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffc4e0 0xffffc4e0
ebp      0x0      0x0

0x804900a <_start+10>  mov    ecx,0x804a000
0x804900f <_start+15>  mov    edx,0x8
0x8049014 <_start+20>  int     0x80
>0x8049016 <_start+22>  mov    eax,0x4
0x804901b <_start+27>  mov    ebx,0x1
0x8049020 <_start+32>  mov    ecx,0x804a008

native process 4800 In: _start      L14    PC: 0x8049016
$4 = 0x804a000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) █
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменили значение регистра ebx (рис. 2.14)



```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffc4e0 0xffffc4e0
ebp      0x0      0x0

0x804900a <_start+10>  mov    ecx,0x804a000
0x804900f <_start+15>  mov    edx,0x8
0x8049014 <_start+20>  int     0x80
>0x8049016 <_start+22>  mov    eax,0x4
0x804901b <_start+27>  mov    ebx,0x1
0x8049020 <_start+32>  mov    ecx,0x804a008

native process 4800 In: _start L14 PC: 0x8049016
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb) █

```

Рис. 2.14: Вывод значения регистра

5. Скопировали файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создали исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ `-args`. Загрузили исполняемый файл в отладчик, указав аргументы.

Для начала установили точку останова перед первой инструкцией в программе и запустили ее.

Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы lab9-3 и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Посмотрели остальные позиции стека – по адресу `[esp+4]` располагается адрес

в памяти где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12] – второго и т.д. (рис. 2.15)

```
(No debugging symbols found in lab09-3)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/argorobcova/work/arch-pc/lab09/
/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, 0x080490e8 in _start ()
(gdb) x/x $esp
0xfffffc450:    0x00000005
(gdb) x/s *(void**)(esp + 4)
0xfffffc692:    "/afs/.dk.sci.pfu.edu.ru/home/a/r/argorobcova/work/arch-pc/lab09/
lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc6da:    "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc6ec:    "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xfffffc6fd:    "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc6ff:    "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.15: Вывод значения регистра

Объясняем, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] - потому что шаг равен размеру переменной - 4 байтам.

6. Преобразовали программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму. (рис. 2.16), (рис. 2.17)

```

lab09-4.asm      [----]  8 L:[  1+10  11/ 37] *(166 / 382b) 0010 0x00A  [*][X]
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 3(10 + x)',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintf
pop ecx.
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end.
pop eax
call atoi
call proc
add esi,eax

```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Переть 7Поиск 8Удаить 9МенюМС10Выход

Рис. 2.16: Программа lab9-4.asm

```

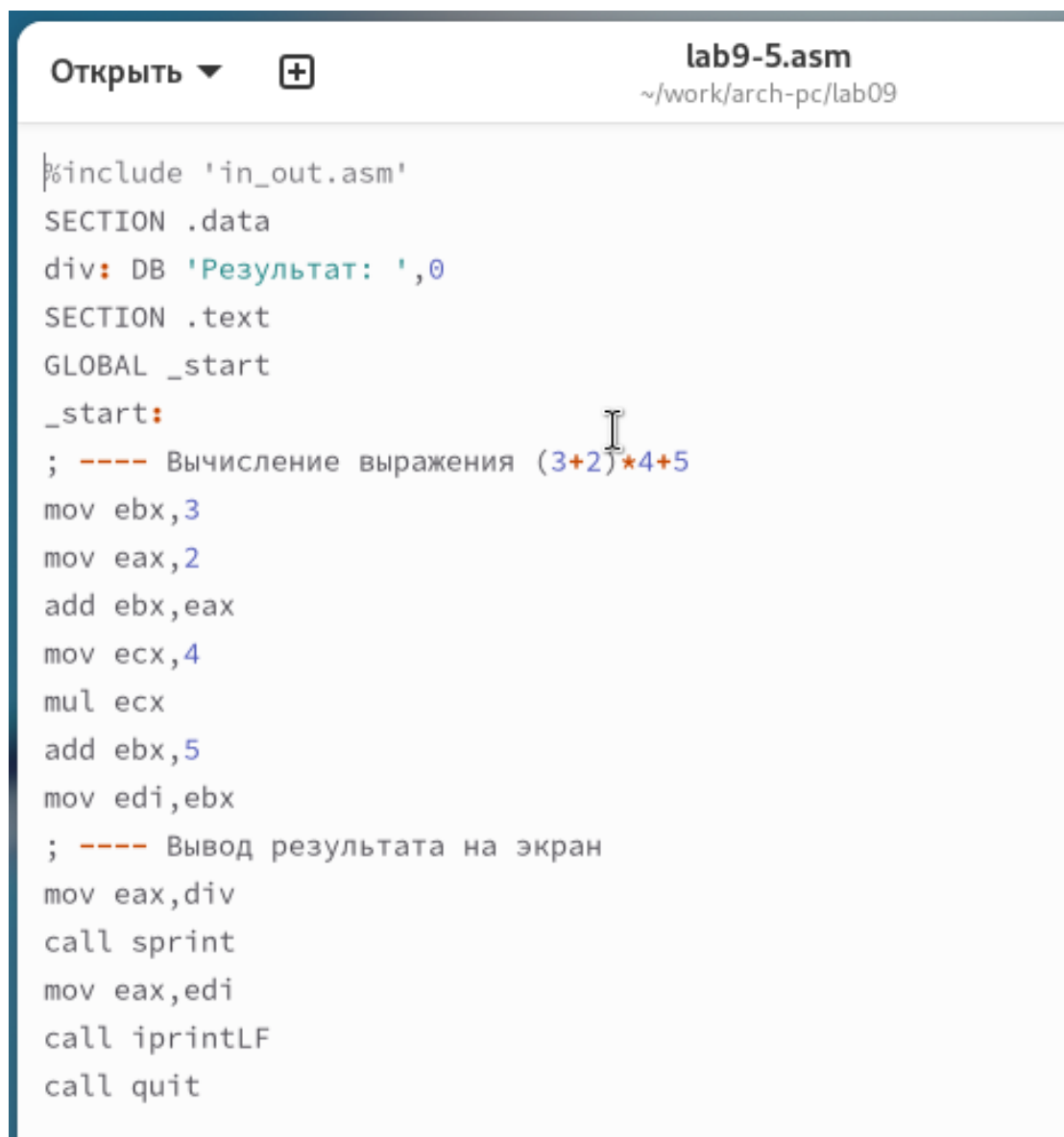
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ nasm -f elf lab09-4.asm
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-4
f(x)= 3(10 + x)
Результат: 0
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-4 2
f(x)= 3(10 + x)
Результат: 36
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-4 1
f(x)= 3(10 + x)
Результат: 33
argorobcova@dk3n35 ~/work/arch-pc/lab09 $ ./lab09-4 1 3 4 6 7
f(x)= 3(10 + x)
Результат: 213
argorobcova@dk3n35 ~/work/arch-pc/lab09 $

```

Рис. 2.17: Запуск программы lab9-4.asm

7. В листинге приведена программа вычисления выражения  $(3+2)*4+5$ . При

запуске данная программа дает неверный результат. Проверили это. С помощью отладчика GDB, анализируя изменения значений регистров, определяем ошибку и исправляем ее. (рис. 2.18)



```
Открыть ▼  + lab9-5.asm
~/work/arch-pc/lab09

#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. 2.18: Код с ошибкой

Заметим, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax

Исправленный код программы (рис. ??)

```
lab09-5.asm      [----]  9 L:[  1+19  20/ 20] *(348 / 348b) <EOF>      [*][X]
#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 iprintLF
call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 2.19: Код исправлен

## **3 Выводы**

Освоили работу с подпрограммами и отладчиком.