

# **Отчёта по лабораторной работе 10**

**Понятие подпрограммы. Отладчик GDB.**

Мягмар Уржиндорж

# 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>28</b>

## Список иллюстраций

2.1	Файл lab10-1.asm . . . . .	7
2.2	Работа программы lab10-1.asm . . . . .	8
2.3	Файл lab10-1.asm . . . . .	9
2.4	Работа программы lab10-1.asm . . . . .	10
2.5	Файл lab10-2.asm . . . . .	11
2.6	Работа программы lab10-2.asm в отладчике . . . . .	12
2.7	дисассимилированный код . . . . .	13
2.8	дисассимилированный код в режиме интел . . . . .	14
2.9	точка остановки . . . . .	15
2.10	изменение регистров . . . . .	16
2.11	изменение регистров . . . . .	17
2.12	изменение значения переменной . . . . .	18
2.13	вывод значения регистра . . . . .	19
2.14	вывод значения регистра . . . . .	20
2.15	вывод значения регистра . . . . .	22
2.16	Файл lab10-4.asm . . . . .	23
2.17	Работа программы lab10-4.asm . . . . .	24
2.18	код с ошибкой . . . . .	25
2.19	отладка . . . . .	26
2.20	код исправлен . . . . .	27
2.21	проверка работы . . . . .	27

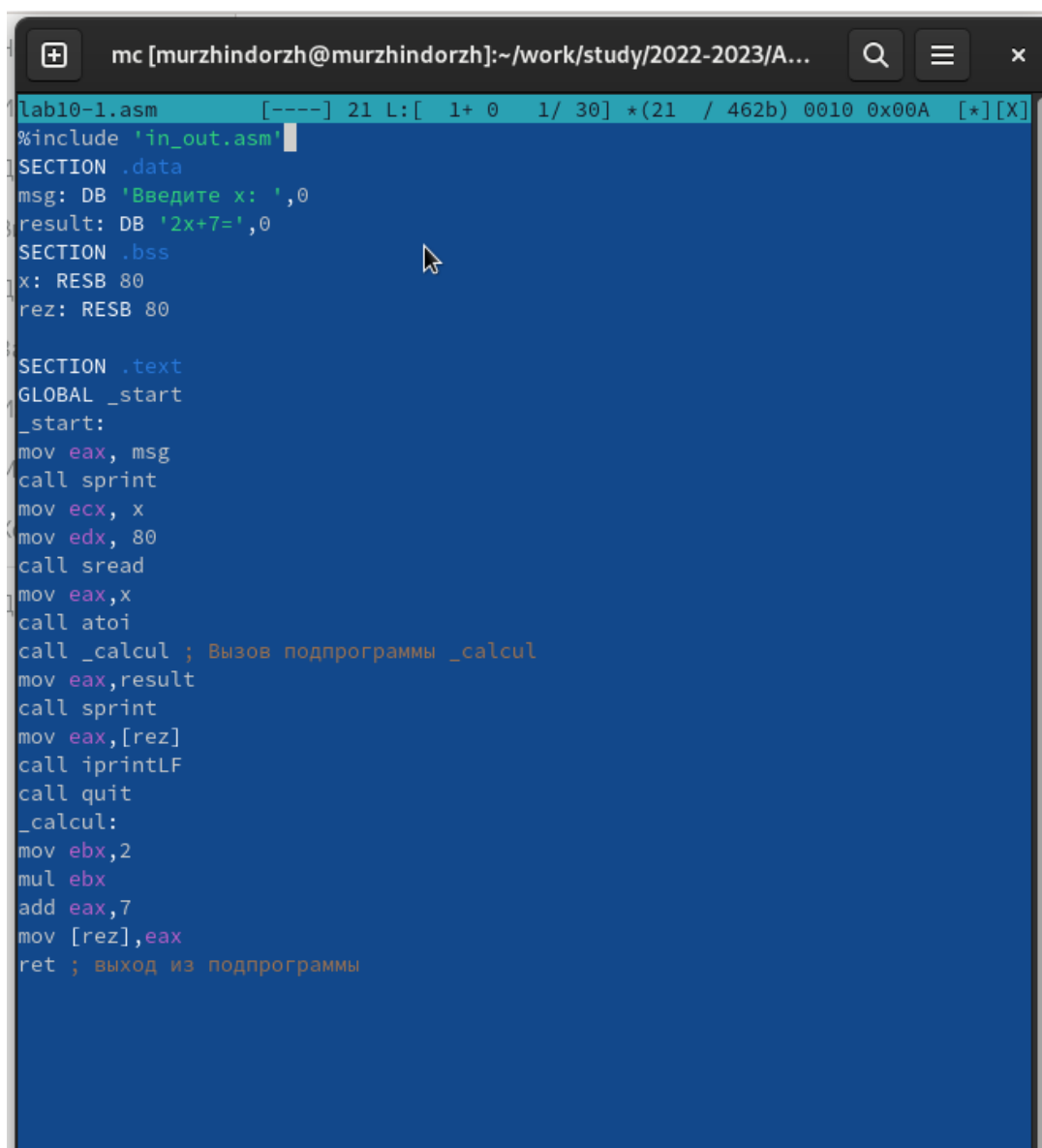
## **Список таблиц**

# 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

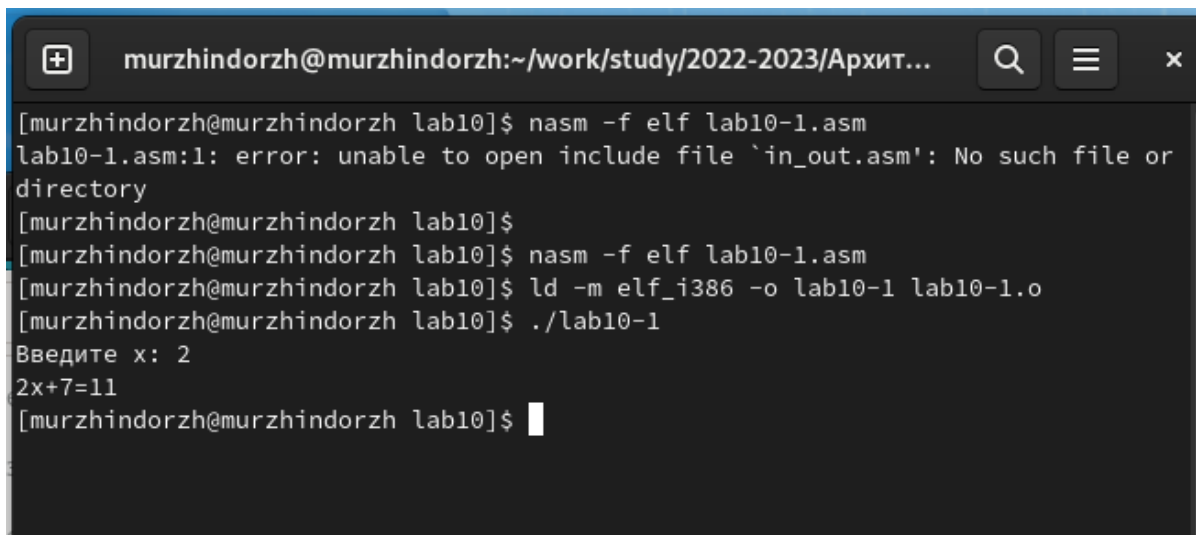
1. Создайте каталог для выполнения лабораторной работы № 10, перейдите в него и создайте файл lab10-1.asm:
2. В качестве примера рассмотрим программу вычисления арифметического выражения  $f(x) = 2x+7$  с помощью подпрограммы calcul. В данном примере  $x$  вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Внимательно изучите текст программы (Листинг 10.1). (рис. [2.1], [2.2])



```
mc [murzhindorzh@murzhindorzh]:~/work/study/2022-2023/A...
lab10-1.asm [----] 21 L: [ 1+ 0 1/ 30] *(21 / 462b) 0010 0x00A [*] [X]
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: 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, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

Рис. 2.1: Файл lab10-1.asm



```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...  
[murzhindorzh@murzhindorzh lab10]$ nasm -f elf lab10-1.asm  
lab10-1.asm:1: error: unable to open include file `in_out.asm': No such file or  
directory  
[murzhindorzh@murzhindorzh lab10]$  
[murzhindorzh@murzhindorzh lab10]$ nasm -f elf lab10-1.asm  
[murzhindorzh@murzhindorzh lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o  
[murzhindorzh@murzhindorzh lab10]$ ./lab10-1  
Введите x: 2  
2x+7=11  
[murzhindorzh@murzhindorzh lab10]$
```

Рис. 2.2: Работа программы lab10-1.asm

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



```
mc [murzhindorzh@murzhindorzh]:~/work/study/2022-2023/A...
lab10-1.asm [----] 9 L: [ 1+35 36/ 40] *(508 / 531b) 0010
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0

SECTION .bss
x: RESB 80
rez: 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, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

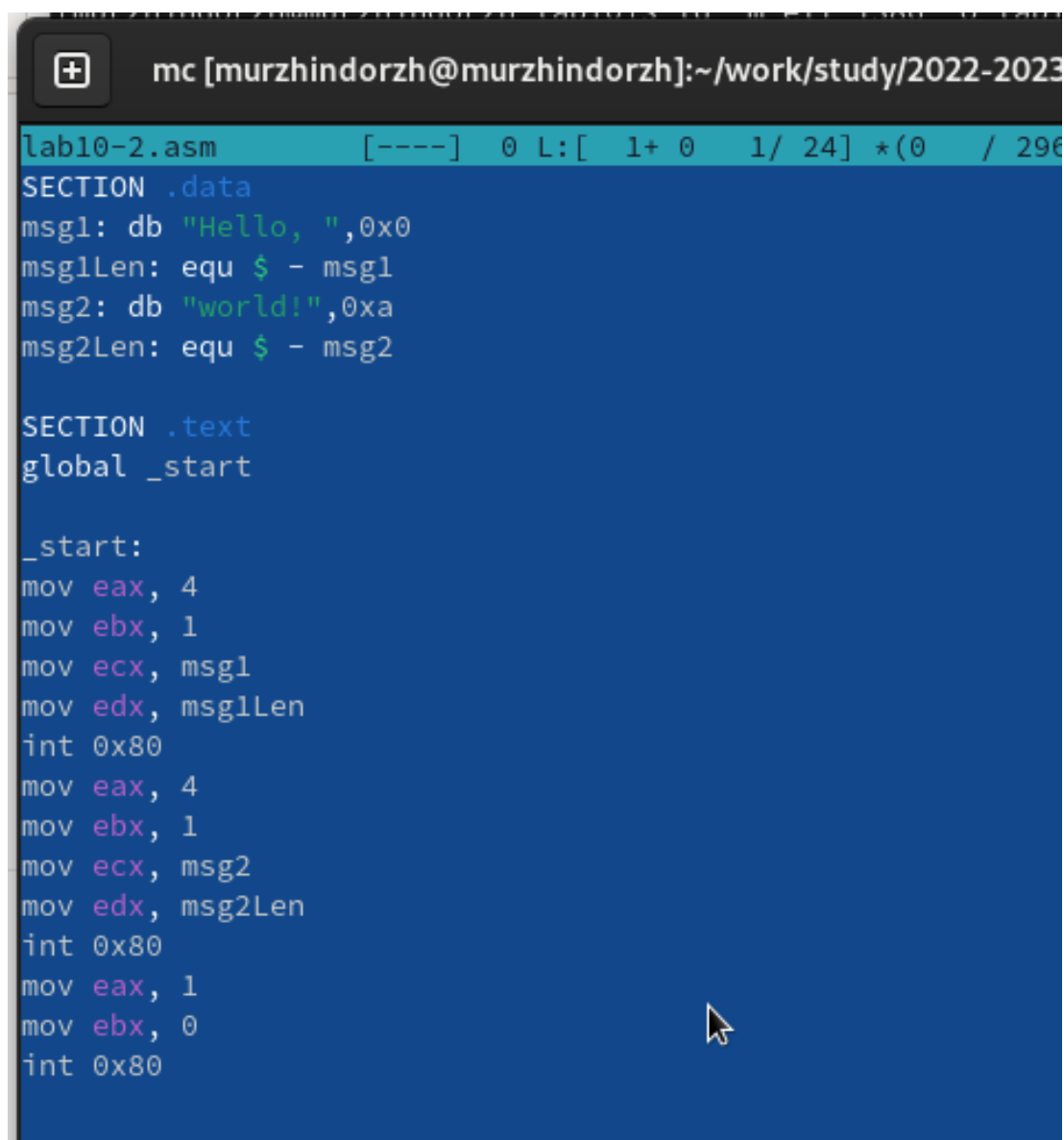
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рис. 2.3: Файл lab10-1.asm

```
[murzhindorzh@murzhindorzh lab10]$  
[murzhindorzh@murzhindorzh lab10]$ nasm -f elf lab10-1.asm  
[murzhindorzh@murzhindorzh lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o  
[murzhindorzh@murzhindorzh lab10]$ ./lab10-1  
Введите x: 2  
2(3x-1)+7=17  
[murzhindorzh@murzhindorzh lab10]$
```

Рис. 2.4: Работа программы lab10-1.asm

4. Создайте файл lab10-2.asm с текстом программы из Листинга 10.2. (Программа печати сообщения Hello world!): (рис. [2.5])



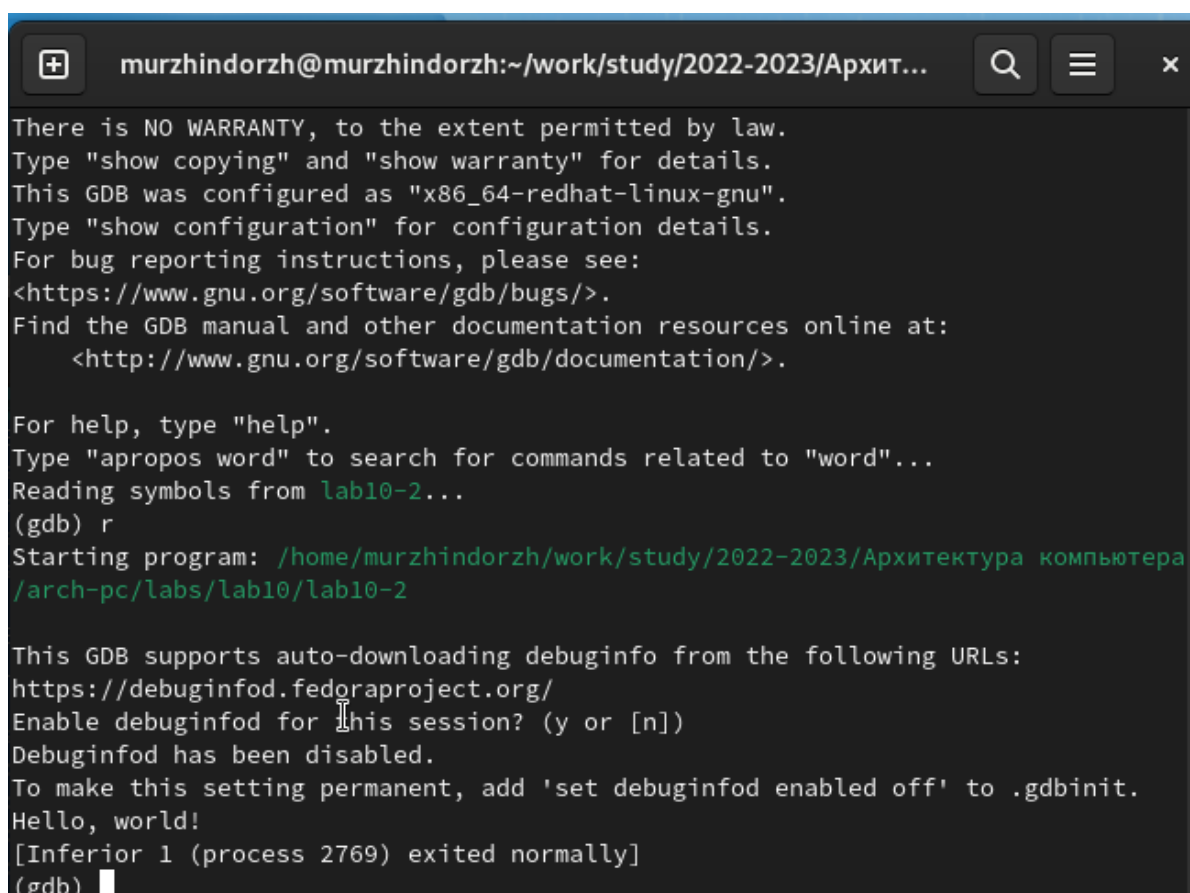
```
mc [murzhindorzh@murzhindorzh]:~/work/study/2022-2023
lab10-2.asm [----] 0 L: [ 1+ 0 1/ 24] *(0 / 296
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2

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. 2.5: Файл lab10-2.asm

Получите исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'. Загрузите исполняемый файл в отладчик gdb: Проверьте работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r):(рис. [2.6])

A screenshot of a terminal window with a dark background. The title bar at the top shows the username 'murzhindorzh' and the current directory path '~ /work/study/2022-2023/Архит...'. The terminal displays the output of a GDB session. It starts with GDB's standard startup messages, including warranty information and configuration details. The user enters the command 'r' to run the program. GDB reports that it is starting the program at the path '/home/murzhindorzh/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab10/lab10-2'. The program outputs 'Hello, world!' and then exits normally. The terminal ends with the GDB prompt '(gdb)' and a cursor.

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...
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) r
Starting program: /home/murzhindorzh/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab10/lab10-2

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 2769) exited normally]
(gdb)
```

Рис. 2.6: Работа программы lab10-2.asm в отладчике

Для более подробного анализа программы установите брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запустите её. Посмотрите дисассимилированный код программы (рис. [2.7], [2.8])

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...
(gdb) run
Starting program: /home/murzhindorzh/work/study/2022-2023/Архитектура ко
/arch-pc/labs/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:11
11      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
      0x0804902a <+42>:     int     $0x80
      0x0804902c <+44>:     mov     $0x1,%eax
      0x08049031 <+49>:     mov     $0x0,%ebx
      0x08049036 <+54>:     int     $0x80
End of assembler dump.
(gdb) 
```

Рис. 2.7: дисассимилированный код

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...
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: дисассимилированный код в режиме интел

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверьте это с помощью команды `info breakpoints` (кратко `i b`) Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определите адрес предпоследней инструкции (`mov ebx,0x0`) и установите точку.(рис. [2.9])

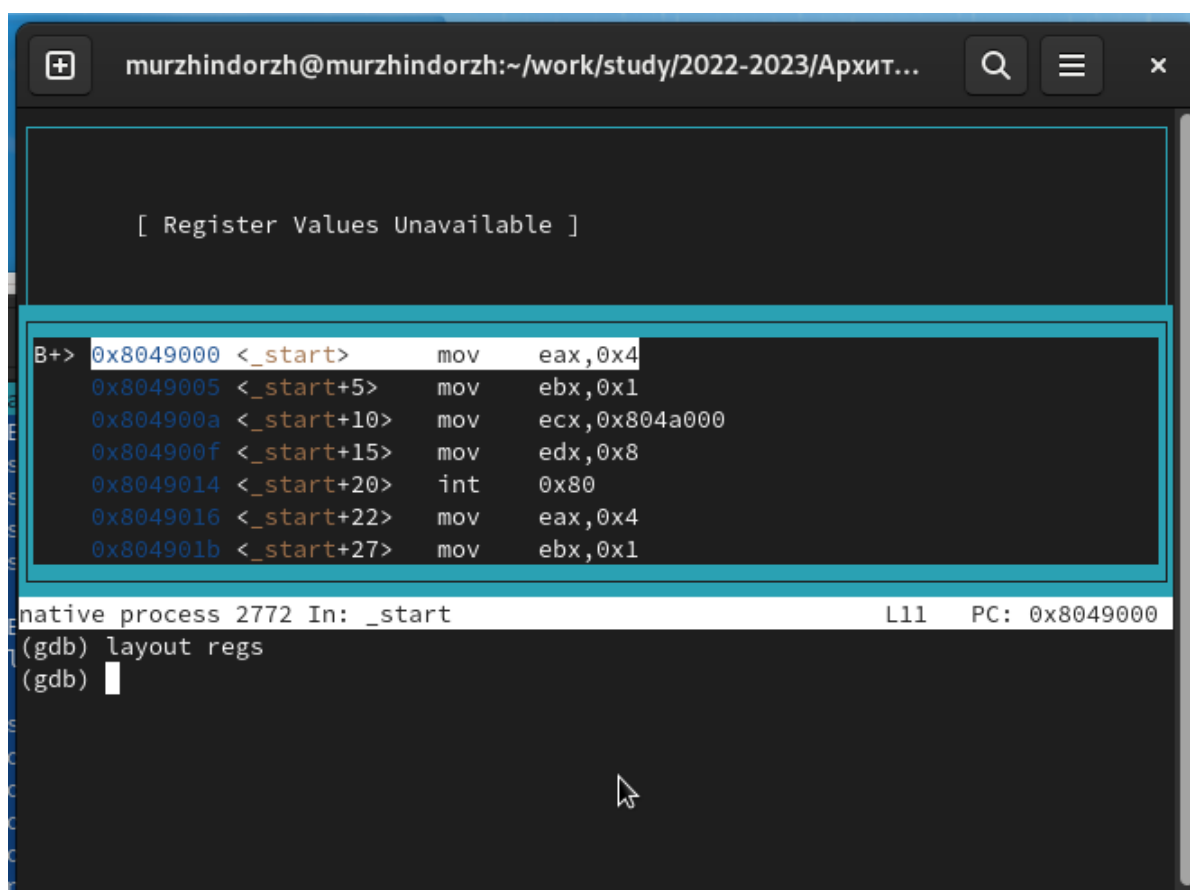


Рис. 2.9: точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполните 5 инструкций с помощью команды `stepi` (или `si`) и проследите за изменением значений регистров. (рис. [2.11] [2.12])

The screenshot shows a GDB terminal window with a dark background. At the top, the window title is "murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...". Below the title bar, there is a message "[ Register Values Unavailable ]". A list of assembly instructions is displayed, each with its address, a comment, and the instruction itself. The instructions are:   
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   
0x804901b <\_start+27> mov ebx,0x1   
Below the list, the status bar shows "native process 2772 In: \_start" and "L11 PC: 0x8049000". At the bottom, the GDB prompt "(gdb)" is followed by the command "layout regs" and a cursor.

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...  
[ 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  
0x804901b <_start+27> mov ebx,0x1  
native process 2772 In: _start L11 PC: 0x8049000  
(gdb) layout regs  
(gdb) █
```

Рис. 2.10: изменение регистров



```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd130 0xffffd130
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x804901b 0x804901b <_start+27>

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
> 0x804901b <_start+27>   mov     ebx,0x1
    0x8049020 <_start+32>  mov     ecx,0x804a008
    0x8049025 <_start+37>  mov     edx,0x7

native process 2772 In: _start L17 PC: 0x0
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
--Type <RET> for more, q to quit, c to continue without paging--sifs
0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.11: изменение регистров

Посмотрите значение переменной msg1 по имени Посмотрите значение переменной msg2 по адресу Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Измените первый символ переменной msg1 Замените любой символ во второй переменной msg2. (рис. [2.12])

The screenshot shows a GDB window with the title bar 'murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...'. The 'Register group: general' panel displays the following values:

Register	Value	Comment
eax	0x4	4
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd130	0xffffd130
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x804901b	0x804901b <_start+27>

The assembly window shows the following code:

```
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
> 0x804901b <_start+27> mov     ebx,0x1
0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7
```

The status bar indicates 'native process 2772 In: \_start L17 PC: 0x804901b'. The GDB console shows the following commands and output:

```
(gdb) si
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lorld!\n\034"
(gdb)
```

Рис. 2.12: изменение значения переменной

Выведете в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx. С помощью команды set измените значение регистра ebx:(рис. [2.13])

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит..  
Register group: general  
eax      0x4      4  
ecx      0x804a000 134520832  
edx      0x8      8  
ebx      0x1      1  
esp      0xffffd130 0xffffd130  
ebp      0x0      0x0  
esi      0x0      0  
edi      0x0      0  
eip      0x804901b 0x804901b <_start+27>  
  
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  
    > 0x804901b <_start+27> mov    ebx,0x1  
    0x8049020 <_start+32> mov    ecx,0x804a008  
    0x8049025 <_start+37> mov    edx,0x7  
  
native process 2772 In: _start L1  
(gdb) p/t $eax  
$2 = 100  
(gdb) p/s $ecx  
$3 = 134520832  
(gdb) p/x $ecx  
$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 измените значение регистра ebx:(рис. [2.14])

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd130 0xffffd130
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x804901b 0x804901b <_start+27>

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
> 0x804901b <_start+27>  mov    ebx,0x1
   0x8049020 <_start+32>  mov    ecx,0x804a008
   0x8049025 <_start+37>  mov    edx,0x7

native process 2772 In: _start L17 PC
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$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. Скопируйте файл lab9-2.asm, созданный при выполнении лабораторной работы №9, с программой выводящей на экран аргументы командной строки. Создайте исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ -args. Загрузите исполняемый файл в отладчик, указав аргументы

Для начала установим точку останова перед первой инструкцией в программе

и запустим 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 `lab10-3` и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Посмотрите 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])

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...
<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...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) r
Starting program: /home/murzhindorzh/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab10/lab10-3 argument 1 argument 2 argument\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab10-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd100: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd2b8: "/home/murzhindorzh/work/study/2022-2023/Архитектура компьютера/
arch-pc/labs/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd327: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd330: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd332: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd33b: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd33d: "argument 3"
(gdb)
```

Рис. 2.15: вывод значения регистра

Объясните, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12]) - шаг равен размеру переменной - 4 байтам.

- Преобразуйте программу из лабораторной работы №9 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму. (рис. [2.16] [2.17])

```

mc [murzhindorzh@murzhindorzh]:~/work/study/20
lab10-4.asm [----] 9 L: [ 1+20 21/ 37] *(24
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)=3(x+2) ',0

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

next:
cmp ecx,0h
jz _end
pop eax
call atoi
call calc
add esi,eax
loop next

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

calc:
add eax,2
mov ebx,3
mul ebx
ret

```

Рис. 2.16: Файл lab10-4.asm

```

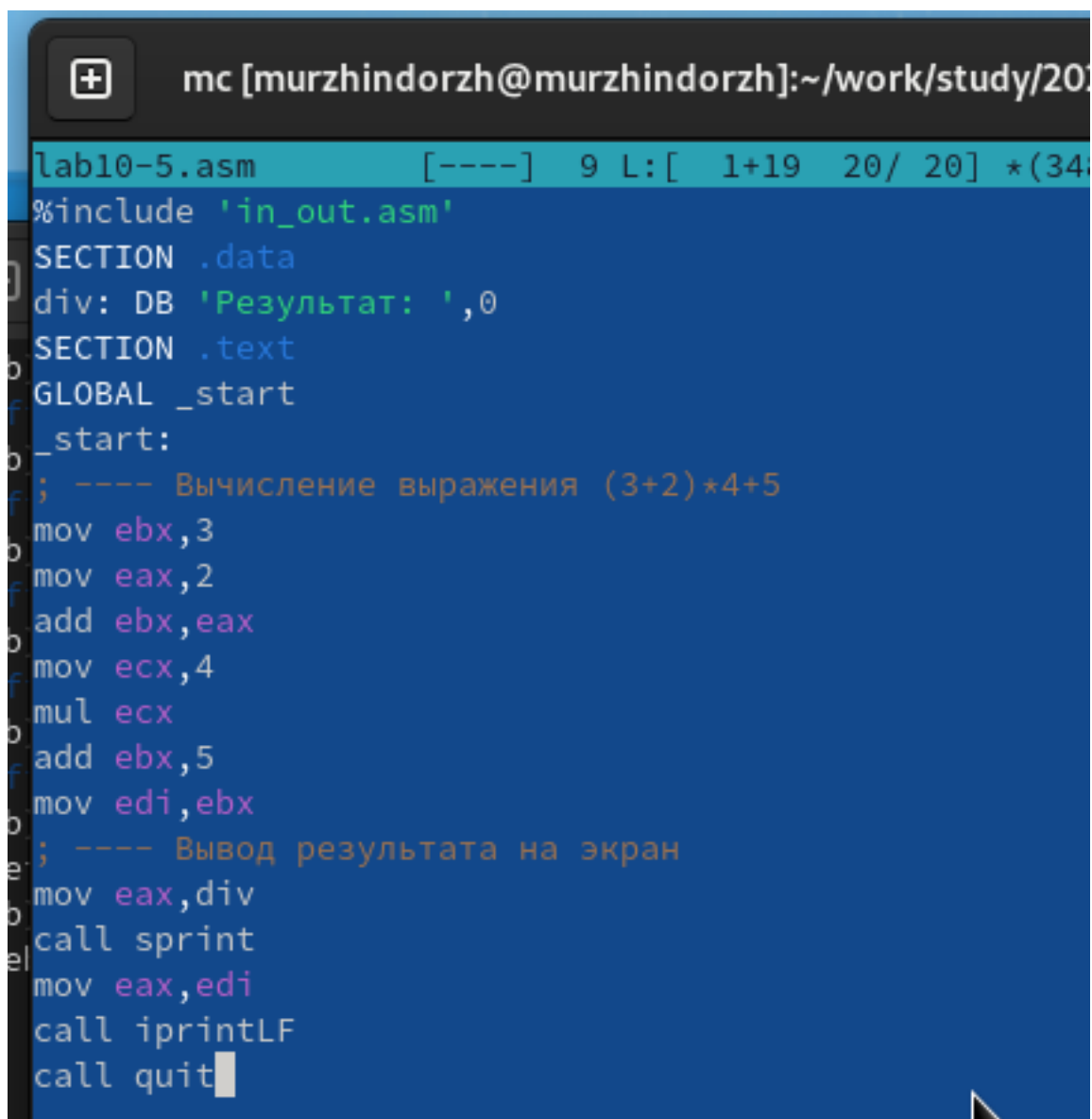
[murzhindorzh@murzhindorzh lab10]$
[murzhindorzh@murzhindorzh lab10]$
[murzhindorzh@murzhindorzh lab10]$ nasm -f elf lab10-4.asm
[murzhindorzh@murzhindorzh lab10]$ ld -m elf_i386 -o lab10-4 lab10-4.o
[murzhindorzh@murzhindorzh lab10]$ ./lab10-4
f(x)=3(x+2)
Результат: 0
[murzhindorzh@murzhindorzh lab10]$ ./lab10-4 1
f(x)=3(x+2)
Результат: 9
[murzhindorzh@murzhindorzh lab10]$ ./lab10-4 1 2 3 4
f(x)=3(x+2)
Результат: 54
[murzhindorzh@murzhindorzh lab10]$

```

Рис. 2.17: Работа программы lab10-4.asm

7. В листинге 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. [2.18] [2.19] [2.20] [2.21])





```
mc [murzhindorzh@murzhindorzh]:~/work/study/20
lab10-5.asm [----] 9 L:[ 1+19 20/ 20] *(34
#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

```
murzhindorzh@murzhindorzh:~/work/study/2022-2023/Архит...

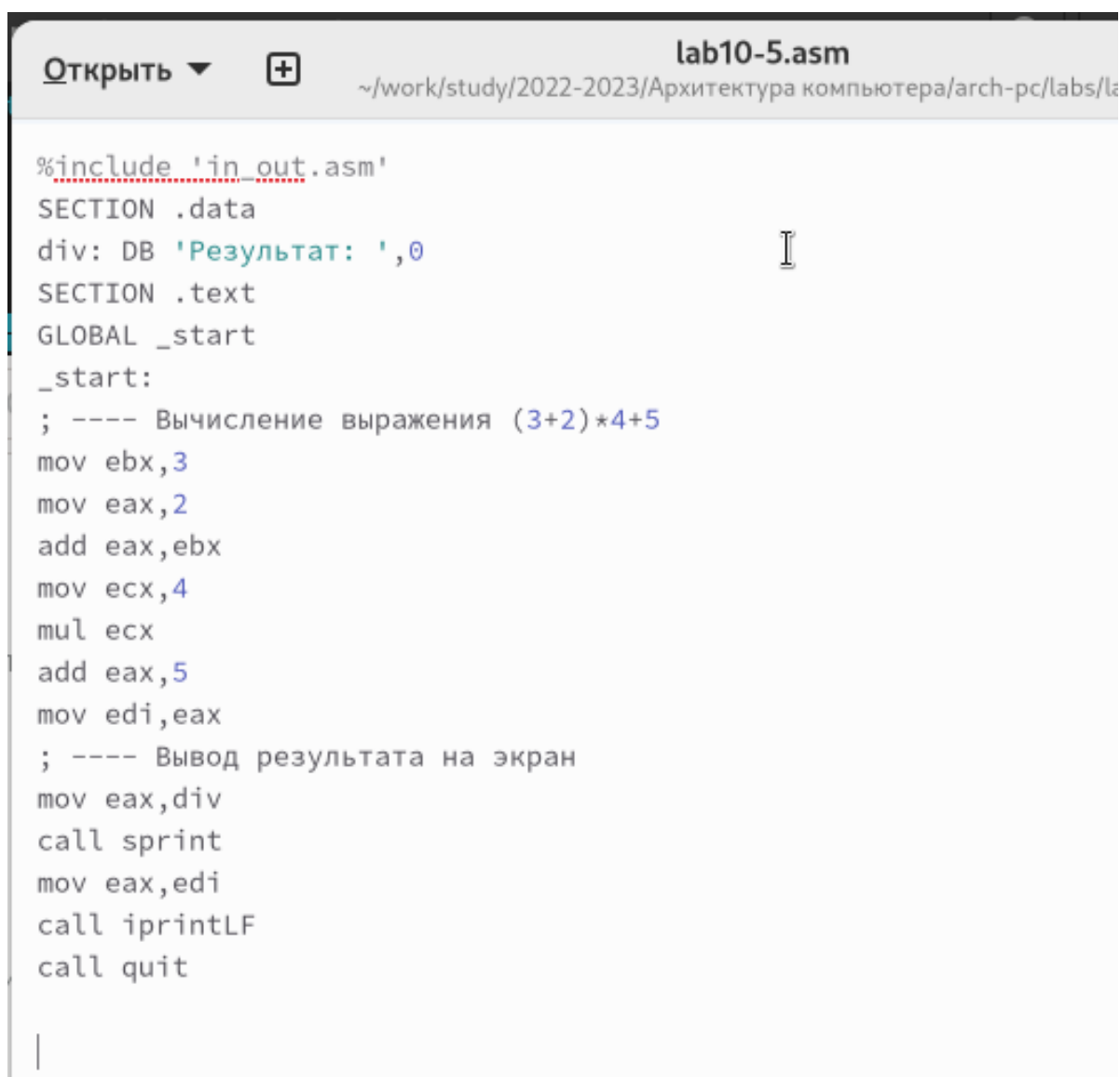
Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd130 0xffffd130

0x80490f9 <_start+17> mul    ecx
0x80490fb <_start+19> add    ebx,0x5
0x80490fe <_start+22> mov    edi,ebx
> 0x8049100 <_start+24> mov    eax,0x804a000
0x8049105 <_start+29> call   0x804900f <sprint>
0x804910a <_start+34> mov    eax,edi
0x804910c <_start+36> call   0x8049086 <iprintLF>

native process 2988 In: _start L16
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.19: отладка

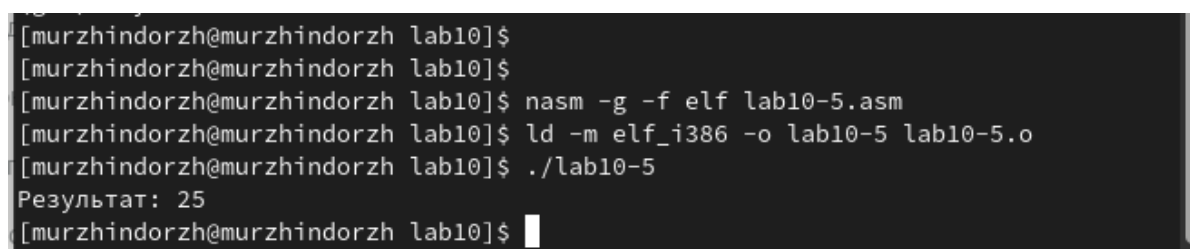
Отметим, что перепутан порядок аргументов у инструкции `add` и что по окончании работы в `edi` отправляется `ebx` вместо `eax`



```
lab10-5.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10-5.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 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. 2.20: код исправлен



```
[murzhindorzh@murzhindorzh lab10]$
[murzhindorzh@murzhindorzh lab10]$
[murzhindorzh@murzhindorzh lab10]$ nasm -g -f elf lab10-5.asm
[murzhindorzh@murzhindorzh lab10]$ ld -m elf_i386 -o lab10-5 lab10-5.o
[murzhindorzh@murzhindorzh lab10]$ ./lab10-5
Результат: 25
[murzhindorzh@murzhindorzh lab10]$
```

Рис. 2.21: проверка работы

## **3 Выводы**

Освоили работу с подпрограммами и отладчиком.