

# **Отчёт по лабораторной работе 9**

**Архитектура компьютеров и операционные системы**

Плетьяго Кирилл НММбд-03-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Самостоятельное задание . . . . .	21
<b>3</b>	<b>Выводы</b>	<b>27</b>

## Список иллюстраций

2.1	Программа в файле lab9-1.asm . . . . .	7
2.2	Запуск программы lab9-1.asm . . . . .	8
2.3	Программа в файле lab9-1.asm . . . . .	9
2.4	Запуск программы lab9-1.asm . . . . .	10
2.5	Программа в файле lab9-2.asm . . . . .	11
2.6	Запуск программы lab9-2.asm в отладчике . . . . .	12
2.7	Дизассемблированный код . . . . .	13
2.8	Дизассемблированный код в режиме интел . . . . .	13
2.9	Точка остановки . . . . .	14
2.10	Изменение регистров . . . . .	15
2.11	Изменение регистров . . . . .	16
2.12	Изменение значения переменной . . . . .	17
2.13	Вывод значения регистра . . . . .	18
2.14	Вывод значения регистра . . . . .	19
2.15	Вывод значения регистра . . . . .	20
2.16	Программа в файле prog-1.asm . . . . .	22
2.17	Запуск программы prog-1.asm . . . . .	23
2.18	Код с ошибкой . . . . .	24
2.19	Отладка . . . . .	25
2.20	Код исправлен . . . . .	26
2.21	Проверка работы . . . . .	26

## 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

Я создал каталог для выполнения лабораторной работы № 9 и перешел в него. Затем я создал файл lab9-1.asm.

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])

```
mc [kapletyago@fedora]:~/work/arch-pc/lab09 [lab9-1.asm [----] 7 L:[ 1+26 27/ 31] *(391 / 463)]
#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: Программа в файле lab9-1.asm

```
[kapletyago@fedora lab09]$  
[kapletyago@fedora lab09]$ nasm -f elf lab9-1.asm  
[kapletyago@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o  
[kapletyago@fedora lab09]$ ./lab9-1  
Введите x: 6  
2x+7=19  
[kapletyago@fedora lab09]$
```

Рис. 2.2: Запуск программы lab9-1.asm

Изменил 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]) (рис. [2.4])



```
mc [kapletyago@fedora]:~/work/arch-pc/lab9-1.asm [----] 0 L: [ 7+33 40/ 40]
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: Программа в файле lab9-1.asm

```
[kapletyago@fedora lab09]$  
[kapletyago@fedora lab09]$ nasm -f elf lab9-1.asm  
[kapletyago@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o  
[kapletyago@fedora lab09]$ ./lab9-1  
Введите x: 6  
2(3x-1)+7=41  
[kapletyago@fedora lab09]$
```

Рис. 2.4: Запуск программы lab9-1.asm

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



```
mc [kapletyago@fedora]:~/work/arch-pc/
lab9-2.asm [----] 8 L: [ 1+19 20/
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2

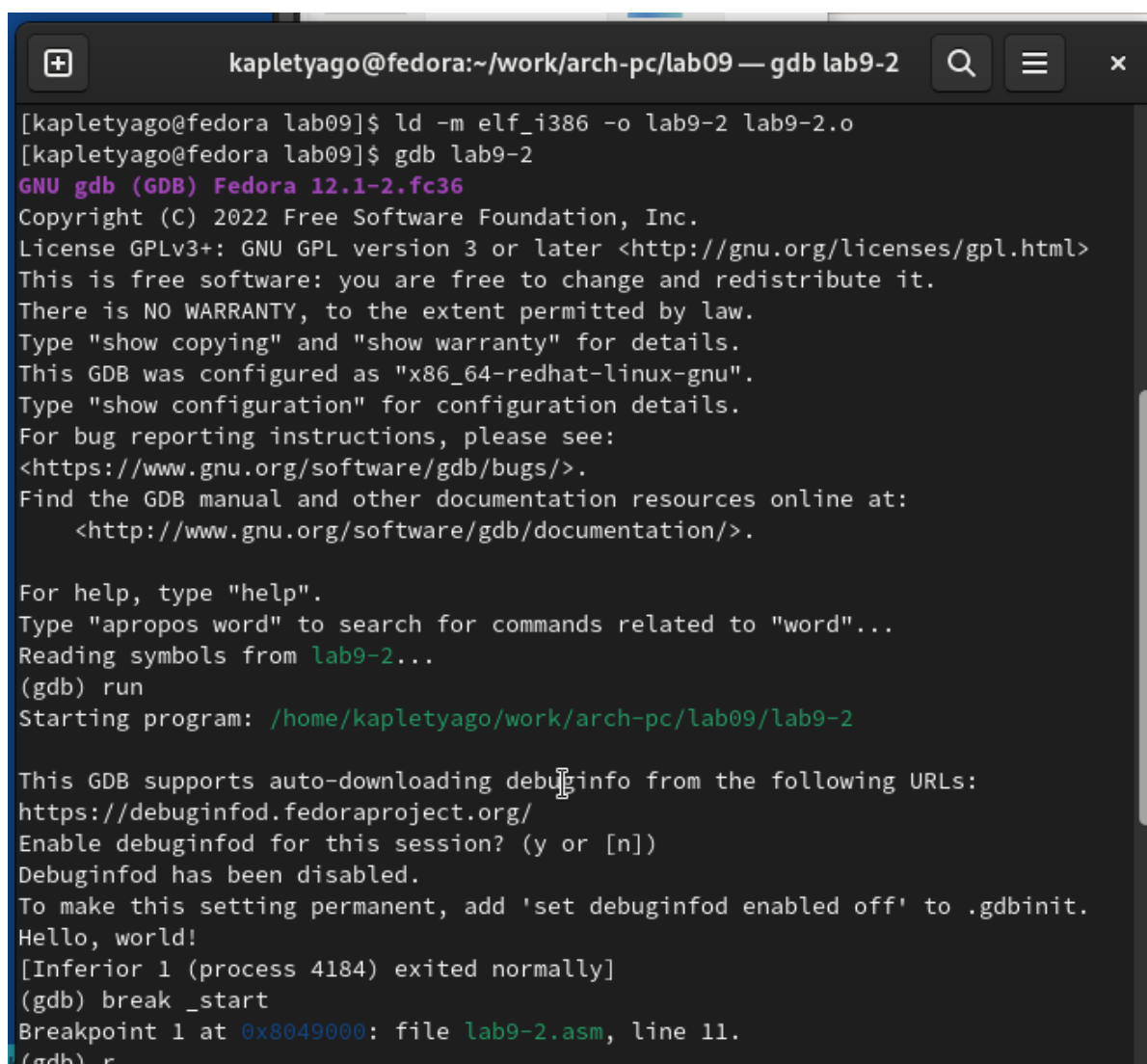
SECTION .text
global _start

_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: Программа в файле lab9-2.asm

Получил исполняемый файл и добавил отладочную информацию с помощью ключа '-g' для работы с GDB.

Загрузил исполняемый файл в отладчик GDB и проверил работу программы, запустив ее с помощью команды 'run' (сокращенно 'r'). (рис. [2.6])

A screenshot of a terminal window titled 'kapletyago@fedora:~/work/arch-pc/lab09 — gdb lab9-2'. The terminal shows the following commands and output:

```
[kapletyago@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[kapletyago@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora 12.1-2.fc36
Copyright (C) 2022 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 lab9-2...
(gdb) run
Starting program: /home/kapletyago/work/arch-pc/lab09/lab9-2

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 4184) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
```

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

Для более подробного анализа программы, установил точку остановки на метке 'start', с которой начинается выполнение любой ассемблерной программы, и запустил ее. Затем просмотрел дизассемблированный код программы.(рис. [2.7]) (рис. [2.8])

```

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.

```

Рис. 2.7: Дизассемблированный код

```

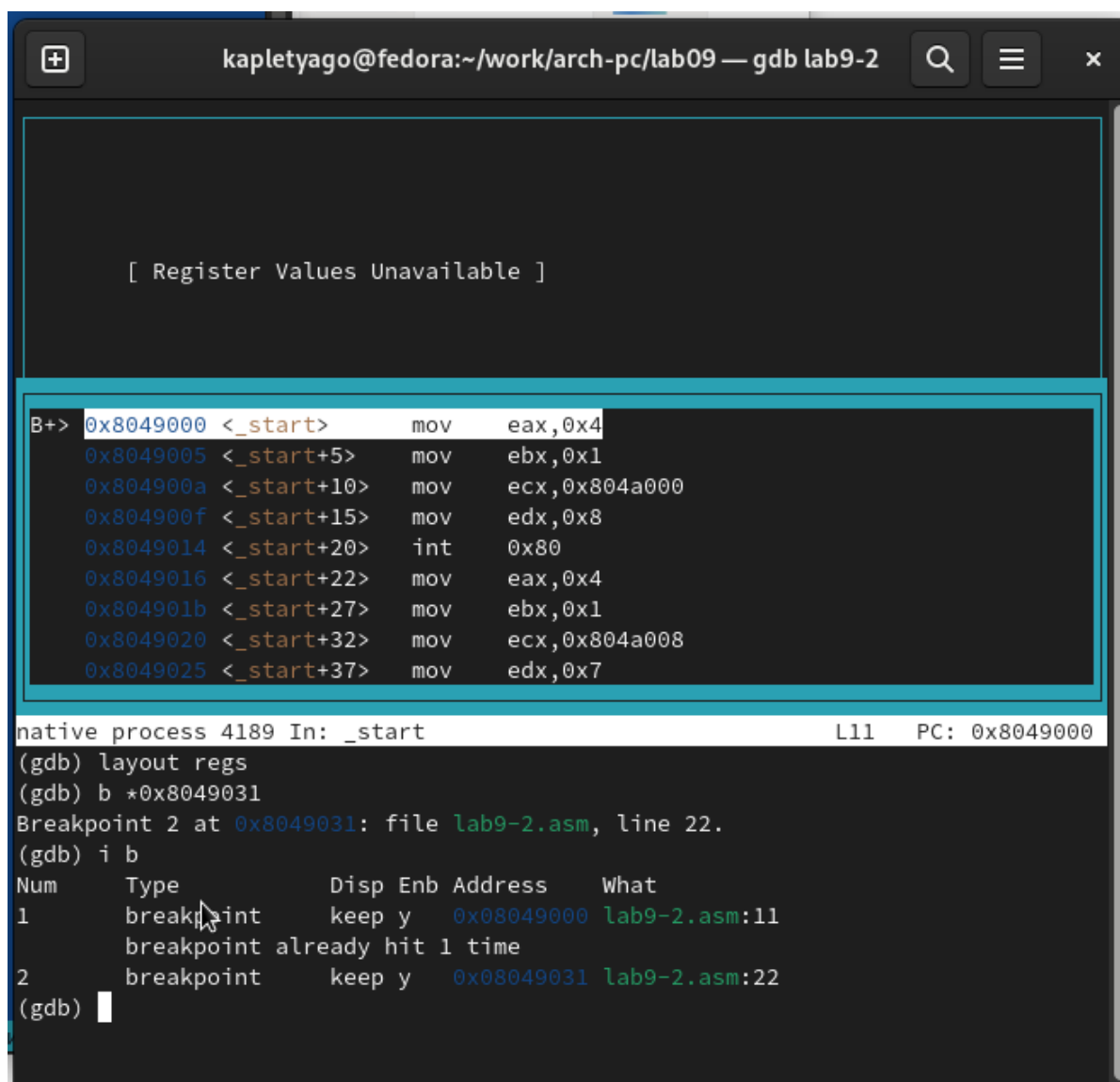
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
    0x08049005 <+5>:      mov     ebx,0x1
    0x0804900a <+10>:     mov     ecx,0x804a000
    0x0804900f <+15>:     mov     edx,0x8
    0x08049014 <+20>:     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

Для проверки точки остановки по имени метки '\_start', использовал команду

‘info breakpoints’ (сокращенно ‘i b’). Затем установил еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции ‘mov ebx, 0x0’. (рис. [2.9])



The screenshot shows a GDB terminal window with the title 'kapletyago@fedora:~/work/arch-pc/lab09 — gdb lab9-2'. The main window displays '[ Register Values Unavailable ]'. Below it, a list of assembly instructions is shown, with the first instruction highlighted:

```
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
```

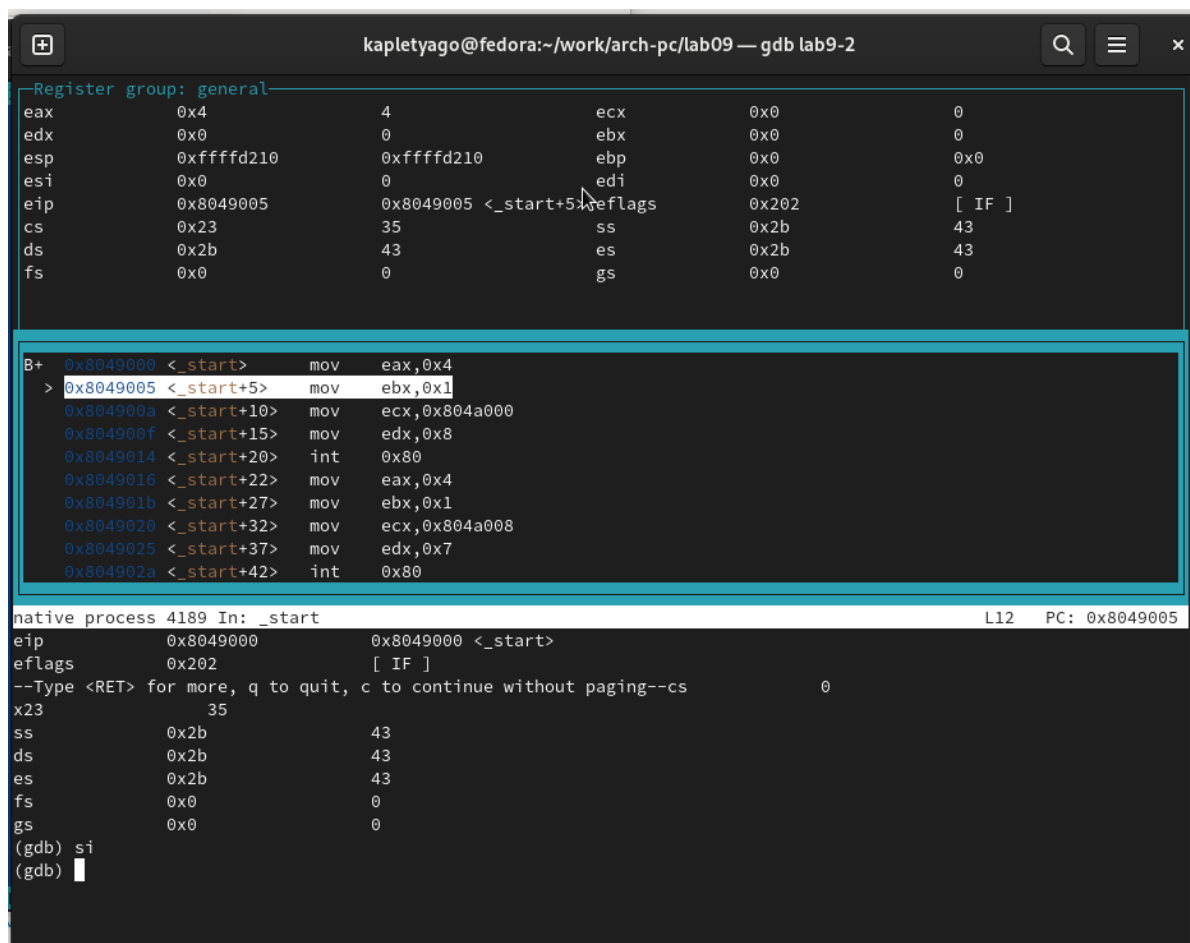
Below the assembly list, the GDB prompt shows the following commands and output:

```
native process 4189 In: _start L11 PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint     keep y   0x08049000 lab9-2.asm:11
       breakpoint already hit 1 time
2      breakpoint     keep y   0x08049031 lab9-2.asm:22
(gdb) 
```

Рис. 2.9: Точка остановки

В отладчике GDB можно просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды ‘stepi’ (сокращенно ‘si’) и отследил изменение значений

регистров. (рис. [2.10]) (рис. [2.11])



The screenshot shows a GDB window titled "kapletyago@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the current values of 16 general-purpose registers in a two-column table. The middle section shows a list of assembly instructions with their corresponding memory addresses and disassembled code. The bottom section displays the state of the native process, including the current instruction pointer (eip), flags, and segment registers.

Register	Value	Register	Value
eax	0x4	ecx	0x0
edx	0x0	ebx	0x0
esp	0xffffd210	ebp	0x0
esi	0x0	edi	0x0
eip	0x8049005	eflags	0x202
cs	0x23	ss	0x2b
ds	0x2b	es	0x2b
fs	0x0	gs	0x0

Address	Disassembly
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
0x804902a <_start+42>	int 0x80

native process 4189 In: \_start L12 PC: 0x8049005

Register	Value
eip	0x8049000 <_start>
eflags	0x202 [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--	cs
x23	35
ss	0x2b
ds	0x2b
es	0x2b
fs	0x0
gs	0x0

(gdb) si  
(gdb) █

Рис. 2.10: Изменение регистров





```
(gdb) si
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
0x804a008 <msg2>:      "world!\n\034"
(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>:      "Lor!d!\n\034"
(gdb)
```

Рис. 2.12: Изменение значения переменной

Для изменения значения регистра или ячейки памяти использовал команду `set`, указав имя регистра или адрес в качестве аргумента. Изменил первый символ переменной `msg1`. (рис. [2.13])

```
native process 4189 In: _start
(gdb) p/s $eax
$1 = 8
$2 = 8
(gdb) p/t $eax
$3 = 1000
(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 изменил значение регистра ebx на нужное значение.  
(рис. [2.14])

```
native process 4189 In: _start
$4 = 0x804a000
(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
7 (gdb) p/s $ebx
$9 = 2
(gdb) 
```

Рис. 2.14: Вывод значения регистра

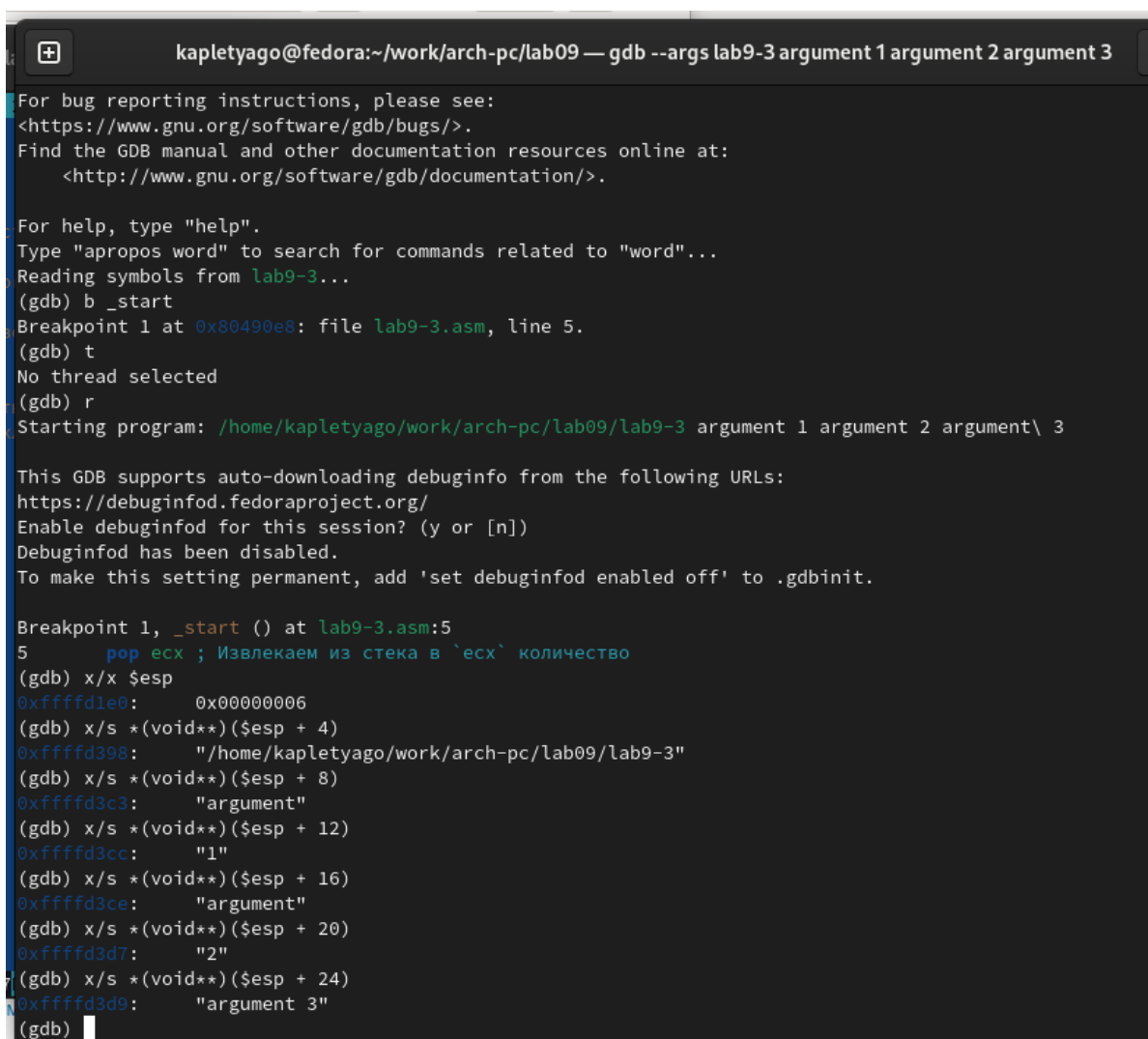
Скопировал файл lab8-2.asm, созданный во время выполнения лабораторной работы №8, который содержит программу для вывода аргументов командной строки. Создал исполняемый файл из скопированного файла.

Для загрузки программы с аргументами в gdb использовал ключ `-args` и загрузил исполняемый файл в отладчик с указанными аргументами.

Установил точку останова перед первой инструкцией программы и запустил ее.

Адрес вершины стека, содержащий количество аргументов командной строки (включая имя программы), хранится в регистре `esp`. По этому адресу находится число, указывающее количество аргументов. В данном случае видно, что количество аргументов равно 5, включая имя программы `lab9-3` и сами аргументы: `аргумент1`, `аргумент2` и `‘аргумент 3’`.

Просмотрел остальные позиции стека. По адресу `[esp+4]` находится адрес в памяти, где располагается имя программы. По адресу `[esp+8]` хранится адрес первого аргумента, по адресу `[esp+12]` - второго и так далее. (рис. [2.15])



```
kapletyago@fedora:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 argument 3
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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) t
No thread selected
(gdb) r
Starting program: /home/kapletyago/work/arch-pc/lab09/lab9-3 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 lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1e0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd398: "/home/kapletyago/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3c3: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3cc: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3ce: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3d7: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3d9: "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего ([esp+4], [esp+8], [esp+12]).

## **2.1 Самостоятельное задание**

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



The image shows a terminal window with a dark background and light blue text. The window title is "mc [kapletyago@fedora]:~/work/arch-pc/lab0". The file being edited is "prog-1.asm", and the cursor is at line 13, column 30. The code is as follows:

```
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 funk
add esi,eax

loop next

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

funk:
mov ebx,2
mul ebx
add eax,7
ret
```

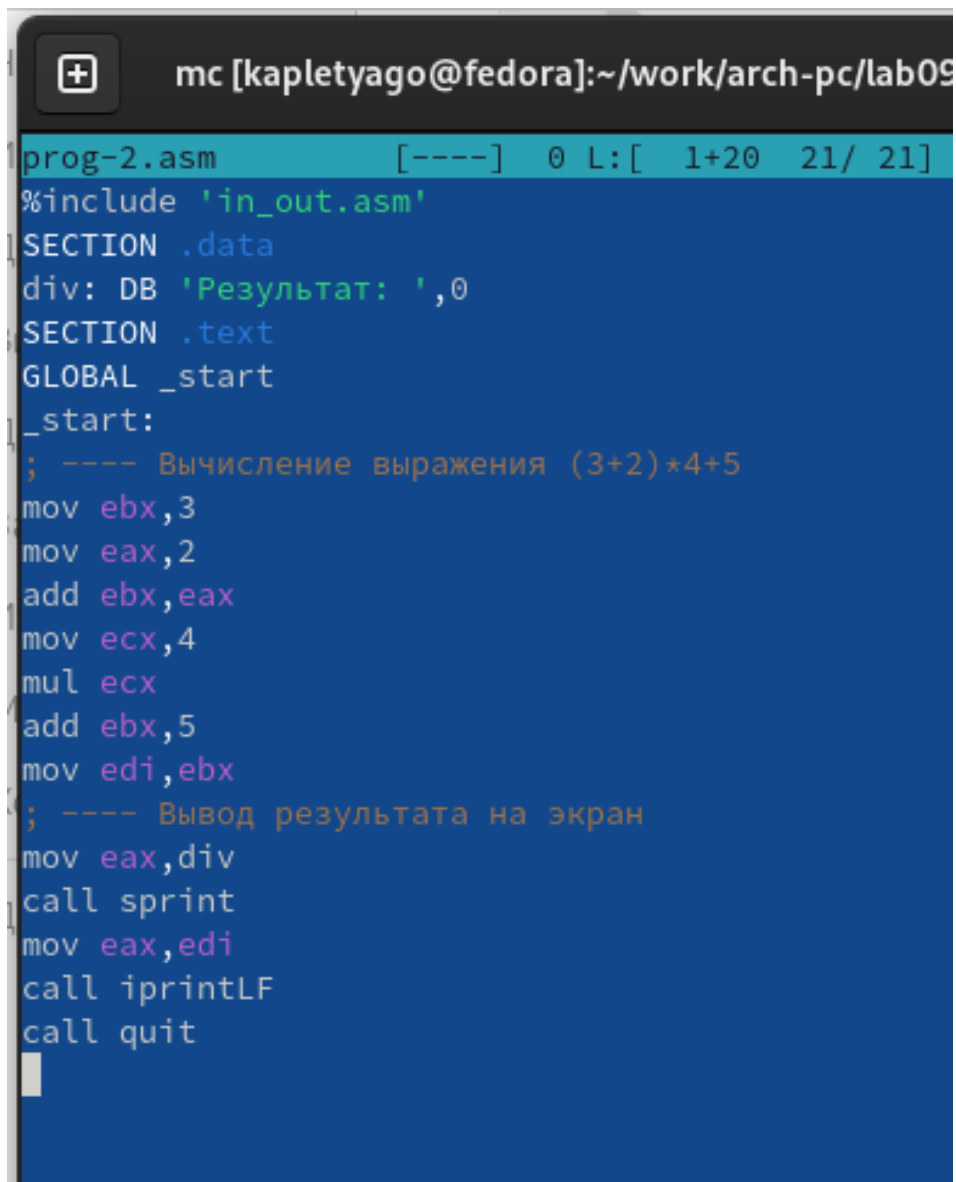
Рис. 2.16: Программа в файле prog-1.asm

```
[kapletyago@fedora lab09]$  
[kapletyago@fedora lab09]$ nasm -f elf prog-1.asm  
[kapletyago@fedora lab09]$ ld -m elf_i386 prog-1.o -o prog-1  
[kapletyago@fedora lab09]$ ./prog-1 1  
f(x)= 7 + 2x  
Результат: 9  
[kapletyago@fedora lab09]$ ./prog-1 1 2 3 4  
f(x)= 7 + 2x  
Результат: 48  
[kapletyago@fedora lab09]$  
[kapletyago@fedora lab09]$
```

Рис. 2.17: Запуск программы prog-1.asm

В листинге приведена программа вычисления выражения  $(3 + 2) * 4 + 5$ . При запуске данная программа дает неверный результат. Проверил это, анализируя изменения значений регистров с помощью отладчика GDB.

Определил ошибку - перепутан порядок аргументов у инструкции add. Также обнаружил, что по окончании работы в edi отправляется ebx вместо eax.(рис. [2.18])



```
mc [kapletyago@fedora]:~/work/arch-pc/lab09
prog-2.asm [----] 0 L:[ 1+20 21/ 21]
#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



```

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

Register group: general
eax      0x8      8      ecx      0x4      4
edx      0x0      0      ebx      0xa     10
esp      0xffffd210 0xffffd210 ebp      0x0      0x0
esi      0x0      0      edi      0xa     10
eip      0x8049100 0x8049100 <_start+24> eflags  0x206    [ PF IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
0x80490f4 <_start+12>   mov     ecx,0x4
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 4876 In: _start L16 PC: 0x8049100
Breakpoint 1 at 0x80490e8: file prog-2.asm, line 8.
(gdb) run
Starting program: /home/kapletyago/work/arch-pc/lab09/prog-2

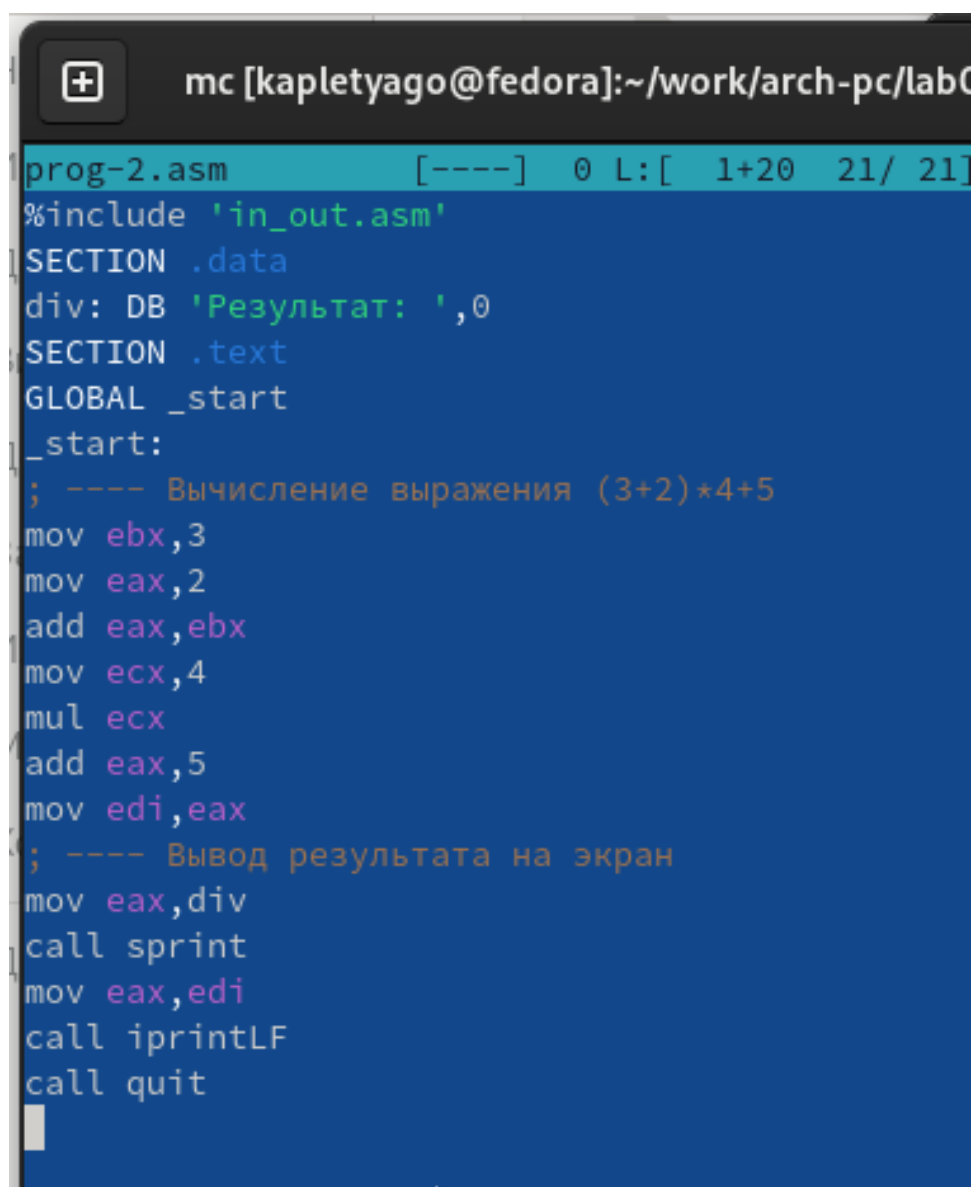
Breakpoint 1, _start () at prog-2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si

```

Рис. 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax (рис. [2.19])

Исправленный код программы (рис. [2.20]) (рис. [2.21])



```
mc [kapletyago@fedora]:~/work/arch-pc/lab09
prog-2.asm [----] 0 L: [ 1+20 21/ 21]
#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



```
[kapletyago@fedora lab09]$
[kapletyago@fedora lab09]$ nasm -f elf prog-2.asm -g
[kapletyago@fedora lab09]$ ld -m elf_i386 prog-2.o -o prog-2
[kapletyago@fedora lab09]$ ./prog-2
Результат: 25
[kapletyago@fedora lab09]$
[kapletyago@fedora lab09]$
```

Рис. 2.21: Проверка работы

## **3 Выводы**

Освоили работу с подпрограммами и отладчиком.