

Отчёт по лабораторной работе 9

Понятие подпрограммы. Отладчик GDB.

Гайбуллаев Фаррух Шухрат

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	33

Список иллюстраций

2.1	Файл lab9-1.asm	7
2.2	Запуск программы lab9-1.asm	9
2.3	Файл lab9-1.asm	10
2.4	Запуск программы lab9-1.asm	12
2.5	Файл lab9-2.asm	13
2.6	Запуск программы lab9-2.asm в отладчике	15
2.7	дисассимилированный код	16
2.8	дисассимилированный код в режиме интел	16
2.9	точка остановки	17
2.10	изменение регистров	18
2.11	изменение регистров	19
2.12	изменение значения переменной	20
2.13	вывод значения регистра	21
2.14	вывод значения регистра	22
2.15	вывод значения регистра	24
2.16	Файл lab9-4.asm	25
2.17	Запуск программы lab9-4.asm	27
2.18	код с ошибкой	28
2.19	отладка	29
2.20	код исправлен	31
2.21	проверка работы	32

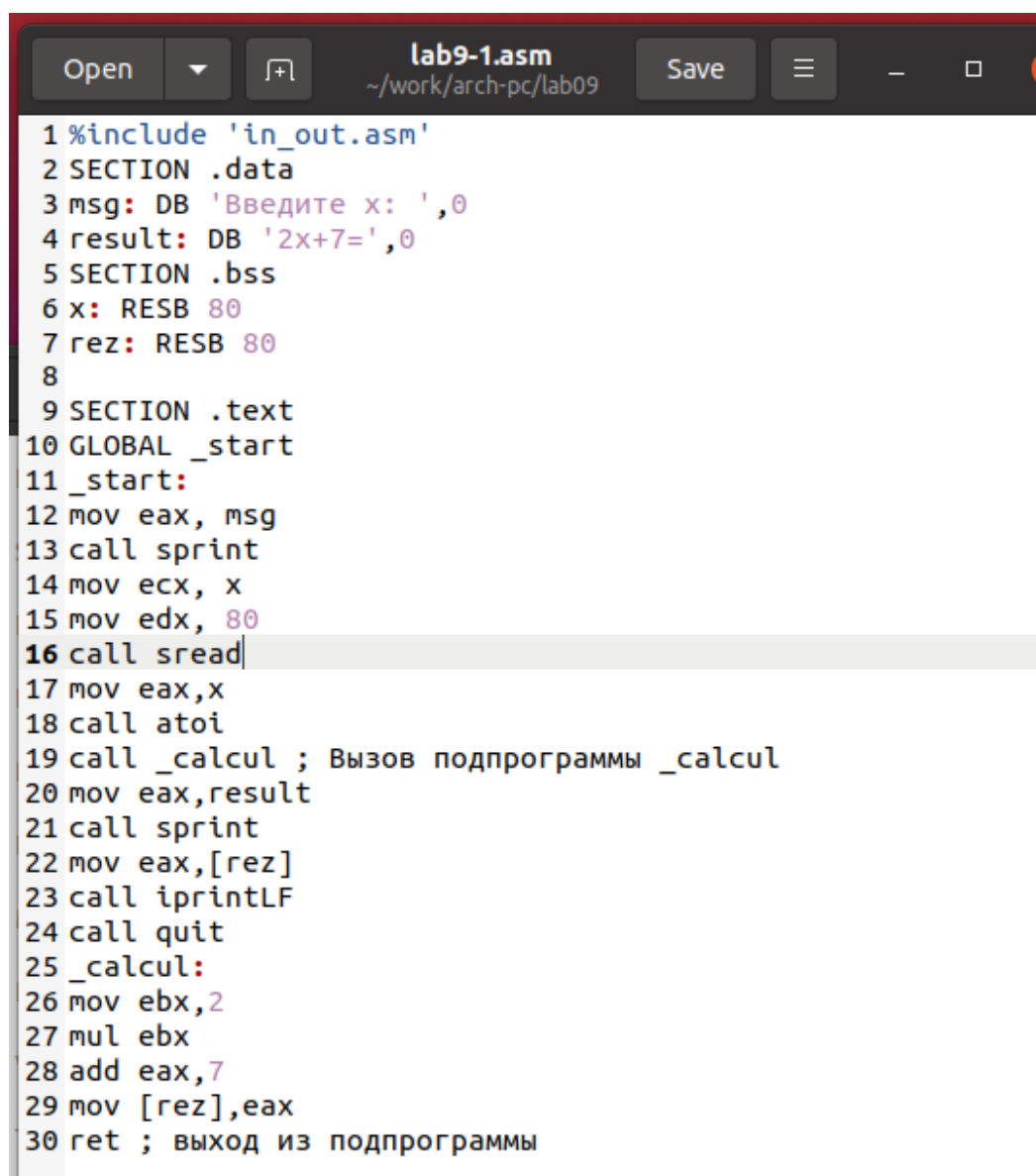
Список таблиц

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 № 9, перейдите в него и создайте 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. Внимательно изучите текст программы (Листинг 10.1).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 rez: RESB 80
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 call _calcul ; Вызов подпрограммы _calcul
20 mov eax, result
21 call sprint
22 mov eax, [rez]
23 call iprintLF
24 call quit
25 _calcul:
26 mov ebx, 2
27 mul ebx
28 add eax, 7
29 mov [rez], eax
30 ret ; выход из подпрограммы
```

Рис. 2.1: Файл lab9-1.asm

```
%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

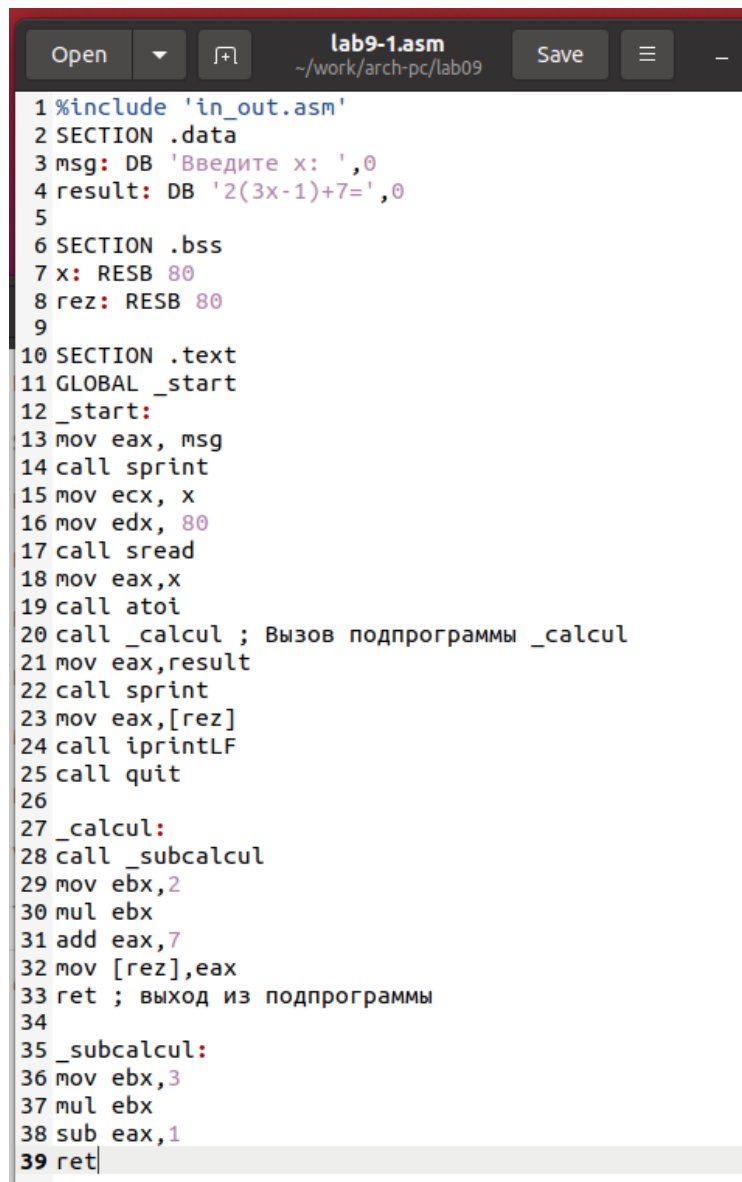
```



```
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2x+7=19
fgaibullaev@ubuntu:~/work/arch-pc/lab09$
```

Рис. 2.2: Запуск программы lab9-1.asm

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



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2(3x-1)+7=',0
5
6 SECTION .bss
7 x: RESB 80
8 rez: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x
19 call atoi
20 call _calcul ; Вызов подпрограммы _calcul
21 mov eax, result
22 call sprint
23 mov eax, [rez]
24 call iprintLF
25 call quit
26
27 _calcul:
28 call _subcalcul
29 mov ebx, 2
30 mul ebx
31 add eax, 7
32 mov [rez], eax
33 ret ; выход из подпрограммы
34
35 _subcalcul:
36 mov ebx, 3
37 mul ebx
38 sub eax, 1
39 ret
```

Рис. 2.3: Файл lab9-1.asm

```
%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

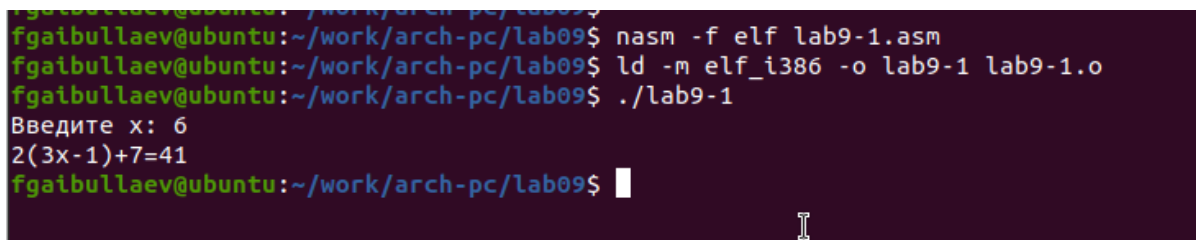
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax,result
call sprint
mov eax,[rez]
call iprintLF
call quit

_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
```

A terminal window with a dark background and green text. The user 'fgaibullaev' is at the 'ubuntu' prompt in the directory '~/work/arch-pc/lab09'. They run 'nasm -f elf lab9-1.asm', then 'ld -m elf_i386 -o lab9-1 lab9-1.o', and finally './lab9-1'. The program prompts 'Введите x: 6', the user enters '6', and the program outputs '2(3x-1)+7=41'.

```
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2(3x-1)+7=41
fgaibullaev@ubuntu:~/work/arch-pc/lab09$
```

Рис. 2.4: Запуск программы lab9-1.asm

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

```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1Len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2Len: equ $ - msg2
6
7 SECTION .text
8 global _start
9
10 _start:
11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msg1Len
15 int 0x80
16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2Len
20 int 0x80
21 mov eax, 1
22 mov ebx, 0
23 int 0x80
```

Рис. 2.5: Файл lab9-2.asm

```
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
```

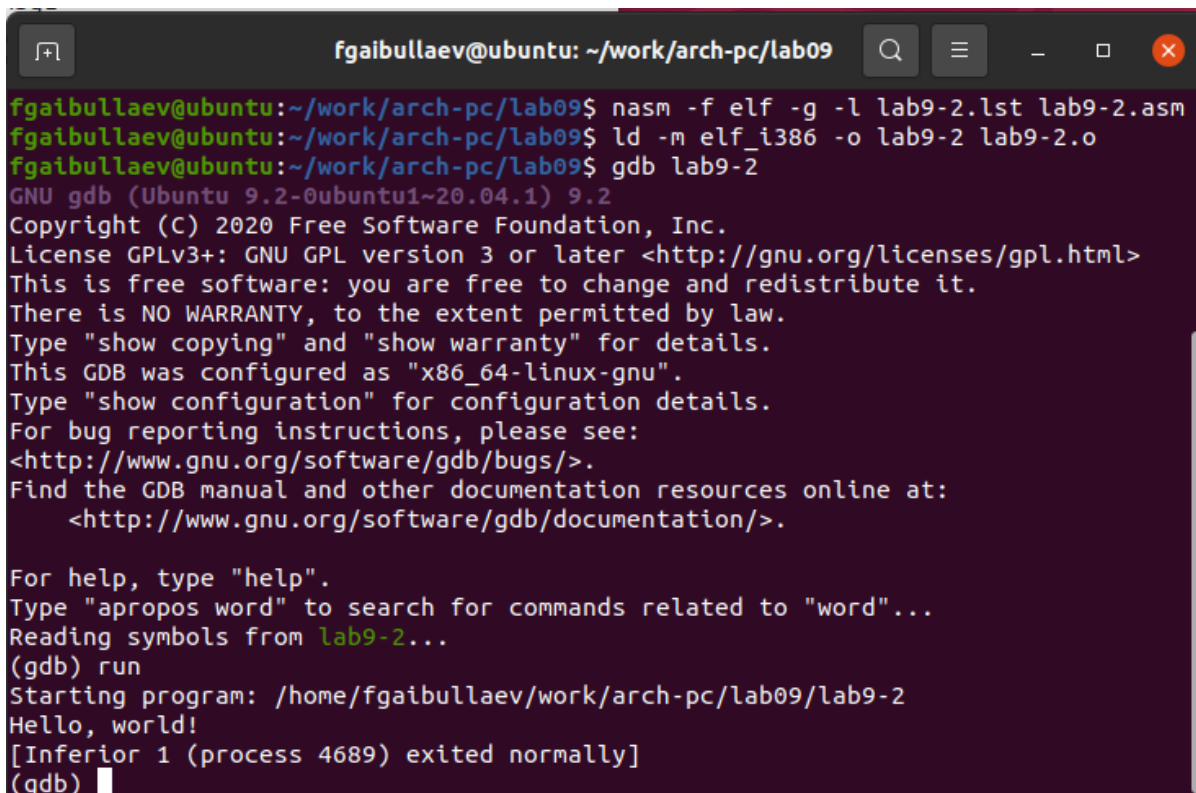
```

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

```

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 terminal window titled 'fgaibullaev@ubuntu: ~/work/arch-pc/lab09' with standard Ubuntu window controls. The terminal shows the following commands and output:

```
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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/fgaibullaev/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 4689) exited normally]
(gdb)
```

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 запустите её. Посмотрите дисассимилированный код программы.

```

(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/fgaibullaev/work/arch-pc/lab09/lab9-2

Breakpoint 1, 0x8049000 in _start ()
(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

```

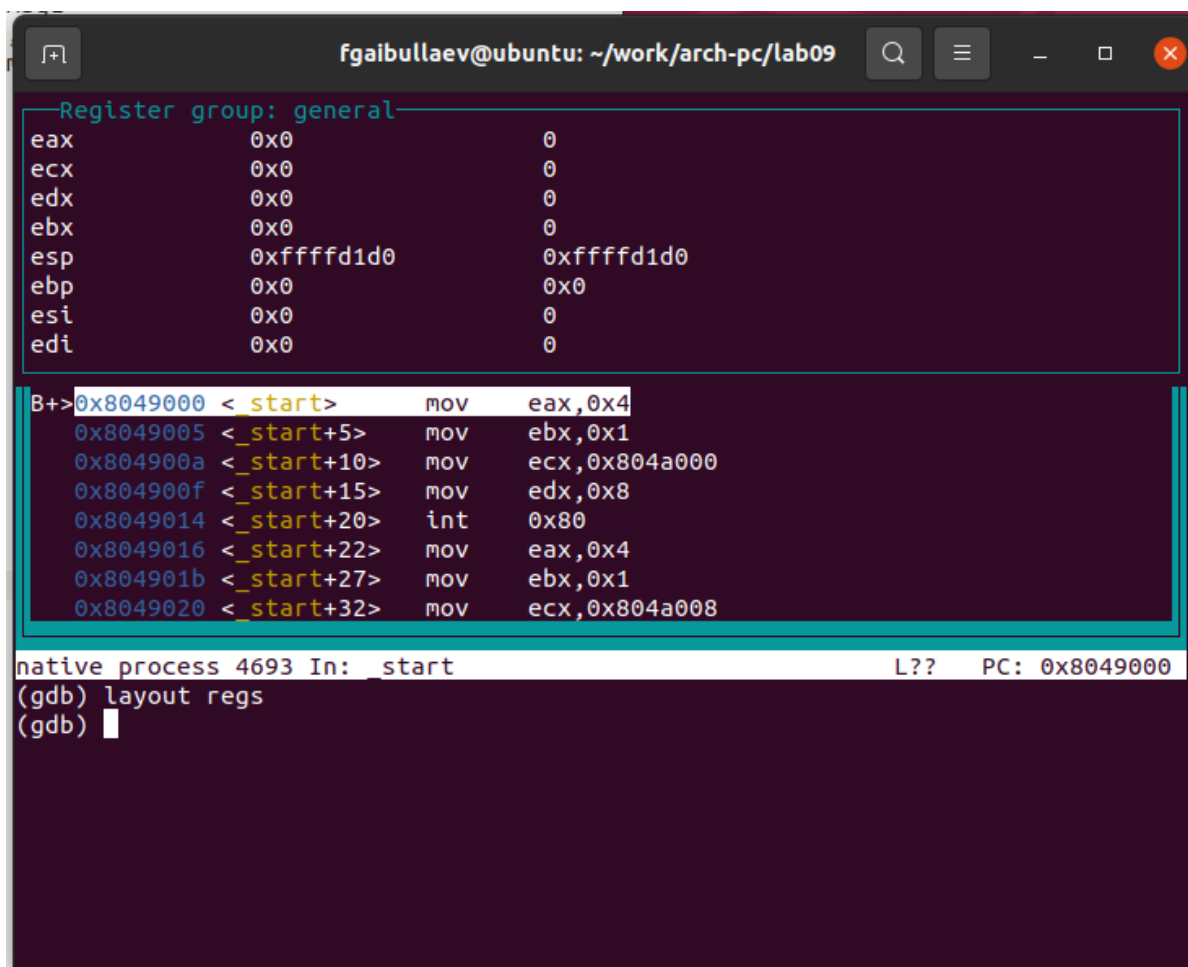
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
    0x08049005 <+5>:      mov     ebx,0x1
    0x0804900a <+10>:     mov     ecx,0x804a000
    0x0804900f <+15>:     mov     edx,0x8
    0x08049014 <+20>:  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



The screenshot shows a GDB terminal window with the title bar "fgaibullaev@ubuntu: ~/work/arch-pc/lab09". The window is divided into two main sections. The top section, titled "Register group: general", displays the values of eight registers: eax (0x0), ecx (0x0), edx (0x0), ebx (0x0), esp (0xffffd1d0), ebp (0x0), esi (0x0), and edi (0x0). The bottom section displays a list of instructions with their addresses and disassembled code. 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; and 0x8049020 <_start+32> mov ecx,0x804a008. The instruction at address 0x804901b is highlighted. At the bottom of the window, the status bar shows "native process 4693 In: _start L?? PC: 0x8049000". The GDB prompt "(gdb) layout regs" is visible, followed by a cursor.

```
fgaibullaev@ubuntu: ~/work/arch-pc/lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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

native process 4693 In: _start L?? PC: 0x8049000
(gdb) layout regs
(gdb) █
```

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.

```
fgaibullaev@ubuntu: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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

native process 4693 In: _start L?? PC: 0x8049005
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
--Type <RET> for more, q to quit, c to continue without paging--si
eflags   0x202      [ IF ]
cs       0x23      35
ss       0x2b      43
ds       0x2b      43
es       0x2b      43
fs       0x0      0
gs       0x0      0
(gdb) si
0x08049005 in _start ()
(gdb)
```

Рис. 2.10: изменение регистров

```
fgaibullaev@ubuntu: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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

native process 4693 In: _start L?? PC: 0x8049016
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
0x08049005 in _start ()
(gdb)
(gdb) si0x0804900a in _start ()
(gdb)
(gdb) si0x0804900f in _start ()
(gdb)
(gdb) si0x08049014 in _start ()
(gdb)
(gdb) si
0x08049016 in _start ()
(gdb)
```

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.

```
fgaibullaev@ubuntu: ~/work/arch-pc/lab09
eax      0x4      4
eax      0x1      1
edx      0x7      7
edx      0x7      7
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

0x804901b <_start+27> mov    ebx,0x1
0x8049020 <_start+32> mov    ecx,0x804a008
0x8049025 <_start+37> mov    edx,0x7
>0x804902a <_start+42> int    0x80
0x804902a <_start+42> int    0x800x1
b+ 0x804902c <_start+44> mov    eax,0x1
B >0x8049031 <_start+49> mov    ebx,0x0
0x8049038      add    BYTE PTR [eax],al

native process 4693 In: _start L?? PC: 0x804902a
(gdb)
world!
0x0804902c in _start ()
(gdb) si

Breakpoint 2, 0x08049031 in _start ()
(gdb)
(gdb) x/1sb &msg10x804a000 <msg1>:      "Hello, "
(gdb)
(gdb) x/1sb 0x804a0080x804a008 <msg2>:  "world!\n"
(gdb)
(gdb)
(gdb) x/1sb &msg10x804a000 <msg1>:      "hello, "
(gdb)
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lorld!\n"
(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:

```
fgaibullaev@ubuntu: ~/work/arch-pc/lab09
eax      0x4      4
eax      0x1      1
edx      0x7      7
edx      0x7      7
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

0x804901b <_start+27> mov     ebx,0x1
0x8049020 <_start+32> mov     ecx,0x804a008
0x8049025 <_start+37> mov     edx,0x7
>0x804902a <_start+42> int     0x80
0x804902a <_start+42> int     0x800x1
b+ 0x804902c <_start+44> mov     eax,0x1
B >0x8049031 <_start+49> mov     ebx,0x0
0x8049038 add     BYTE PTR [eax],al

native process 4693 In: _start L?? PC: 0x804902a
(gdb)
0x804a008 <msg2>: "Lorld!\n"
(gdb)
(gdb) p/s $eax$1 = 1
(gdb)
(gdb) p/t $eax$2 = 1
(gdb)
(gdb) p/s $ecx$3 = 134520840
(gdb)
(gdb) p/x $ecx$4 = 0x804a008
(gdb)
(gdb) p/s $edx$5 = 7
(gdb)
(gdb) p/t $edx$6 = 111
(gdb) p/x $edx
$7 = 0x7
(gdb)
```

Рис. 2.13: вывод значения регистра

С помощью команды set измените значение регистра ebx

```
fgaibullaev@ubuntu: ~/work/arch-pc/lab09
eax      0x4      4
eax      0x1      1
edx      0x7      7
edx      0x7      7
ebx      0x2      2
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

0x804901b <_start+27>  mov     ebx,0x1
0x8049020 <_start+32>  mov     ecx,0x804a008
0x8049025 <_start+37>  mov     edx,0x7
>0x804902a <_start+42>  int     0x80
0x804902a <_start+42>  int     0x800x1
b+ 0x804902c <_start+44>  mov     eax,0x1
B >0x8049031 <_start+49>  mov     ebx,0x0
0x8049038          add     BYTE PTR [eax],al

native process 4693 In: _start L?? PC: 0x804902a
(gdb)
(gdb) p/x $ecx$4 = 0x804a008
(gdb)
(gdb) p/s $edx$5 = 7
(gdb)
(gdb) p/t $edx$6 = 111
(gdb) p/x $edx
$7 = 0x7
$8 = 0x7
$9 = 0x7
(gdb)
(gdb)
(gdb) p/s $ebx$10 = 50
(gdb)
(gdb) p/s $ebx
$11 = 2
(gdb)
```

Рис. 2.14: вывод значения регистра

5. Скопируйте файл lab8-2.asm, созданный при выполнении лабораторной работы №8, 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 lab9-3 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.

```
fgaibullaev@ubuntu: ~/work/arch-pc/lab09
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ gdb --args lab9-3 argument 1 argument 2
'argument 3'
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

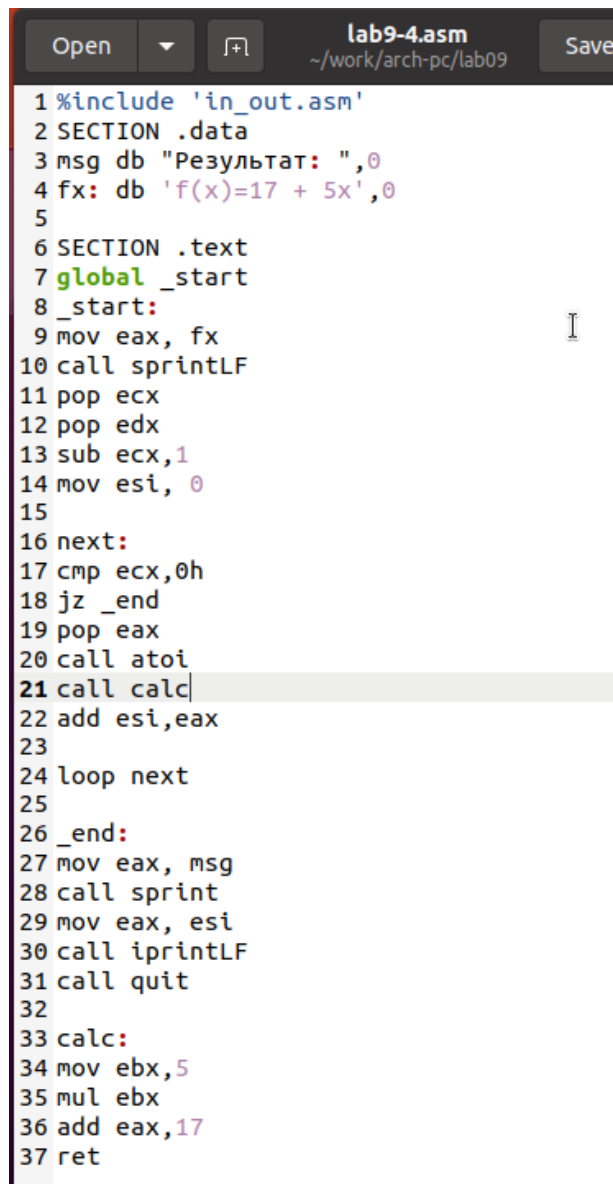
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x080490e8
(gdb) run
Starting program: /home/fgaibullaev/work/arch-pc/lab09/lab9-3 argument 1 argumen
t 2 argument\ 3

Breakpoint 1, 0x080490e8 in _start ()
(gdb) x/x $esp
0xffffd190: 0x00000006
(gdb) x/s *(void**)($esp + 4)
0xffffd35a: "/home/fgaibullaev/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd386: "argument"
(gdb) x/s *(void**)($esp + 12)
0xffffd38f: "1"
(gdb) x/s *(void**)($esp + 16)
0xffffd391: "argument"
(gdb) x/s *(void**)($esp + 20)
0xffffd39a: "2"
(gdb) x/s *(void**)($esp + 24)
0xffffd39c: "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 №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)=17 + 5x',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 call calc
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprint
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 calc:
34 mov ebx,5
35 mul ebx
36 add eax,17
37 ret
```

Рис. 2.16: Файл lab9-4.asm

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0
fx: db 'f(x)=17 + 5x',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

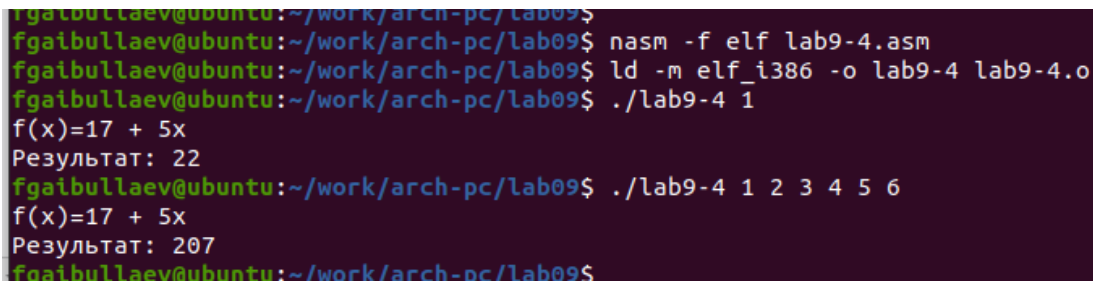
loop next

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

calc:
mov ebx,5
mul ebx

```

```
add eax,17
ret
```



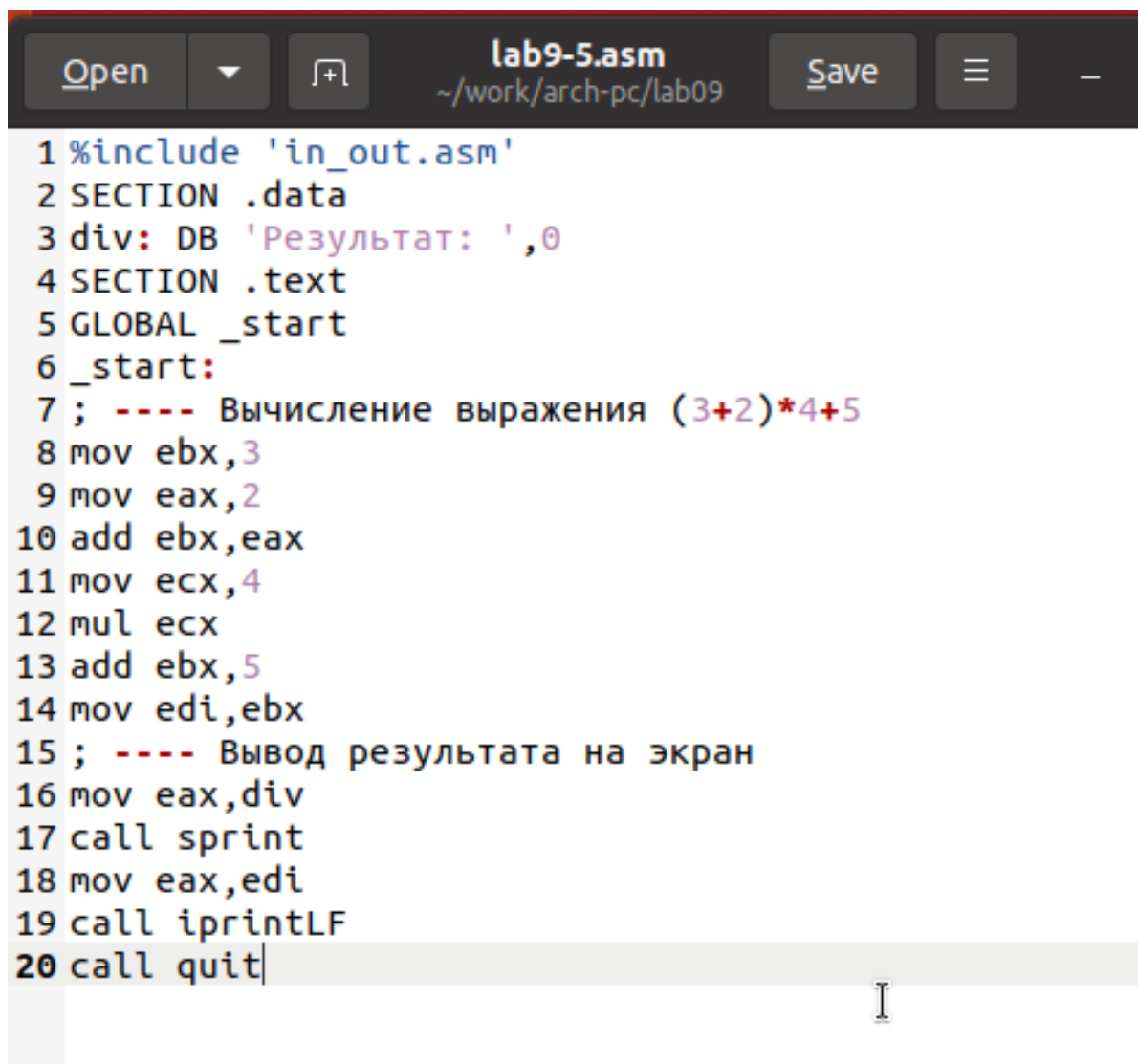
```
fgaibullaev@ubuntu:~/work/arch-pc/lab09$
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ./lab9-4 1
f(x)=17 + 5x
Результат: 22
fgaibullaev@ubuntu:~/work/arch-pc/lab09$ ./lab9-4 1 2 3 4 5 6
f(x)=17 + 5x
Результат: 207
fgaibullaev@ubuntu:~/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. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.

```
%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
```



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

1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ---- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add ebx,eax
11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
```

Рис. 2.18: код с ошибкой

```

fgaibullaev@ubuntu: ~/work/arch-pc/lab09
Register group: general
eax      0x2      2
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f9 0x80490f9 <_start+17>
eflags   0x206    [ PF IF ]
cs       0x23     35
ss       0x2b     43

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>
   0x8049111 <_start+41> call    0x80490db <quit>

native process 4756 In: _start L?? PC: 0x80490f9
(gdb) b _startBreakpoint 1 at 0x80490e8
(gdb)
(gdb) runStarting program: /home/fgaibullaev/work/arch-pc/lab09/lab9-5

Breakpoint 1, 0x080490e8 in _start ()
(gdb)
(gdb) si0x080490ed in _start ()
(gdb)
(gdb) si0x080490f2 in _start ()
(gdb)
(gdb) si0x080490f4 in _start ()
(gdb) si
0x080490f9 in _start ()
(gdb)

```

Рис. 2.19: отладка

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

```

#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

SECTION .text

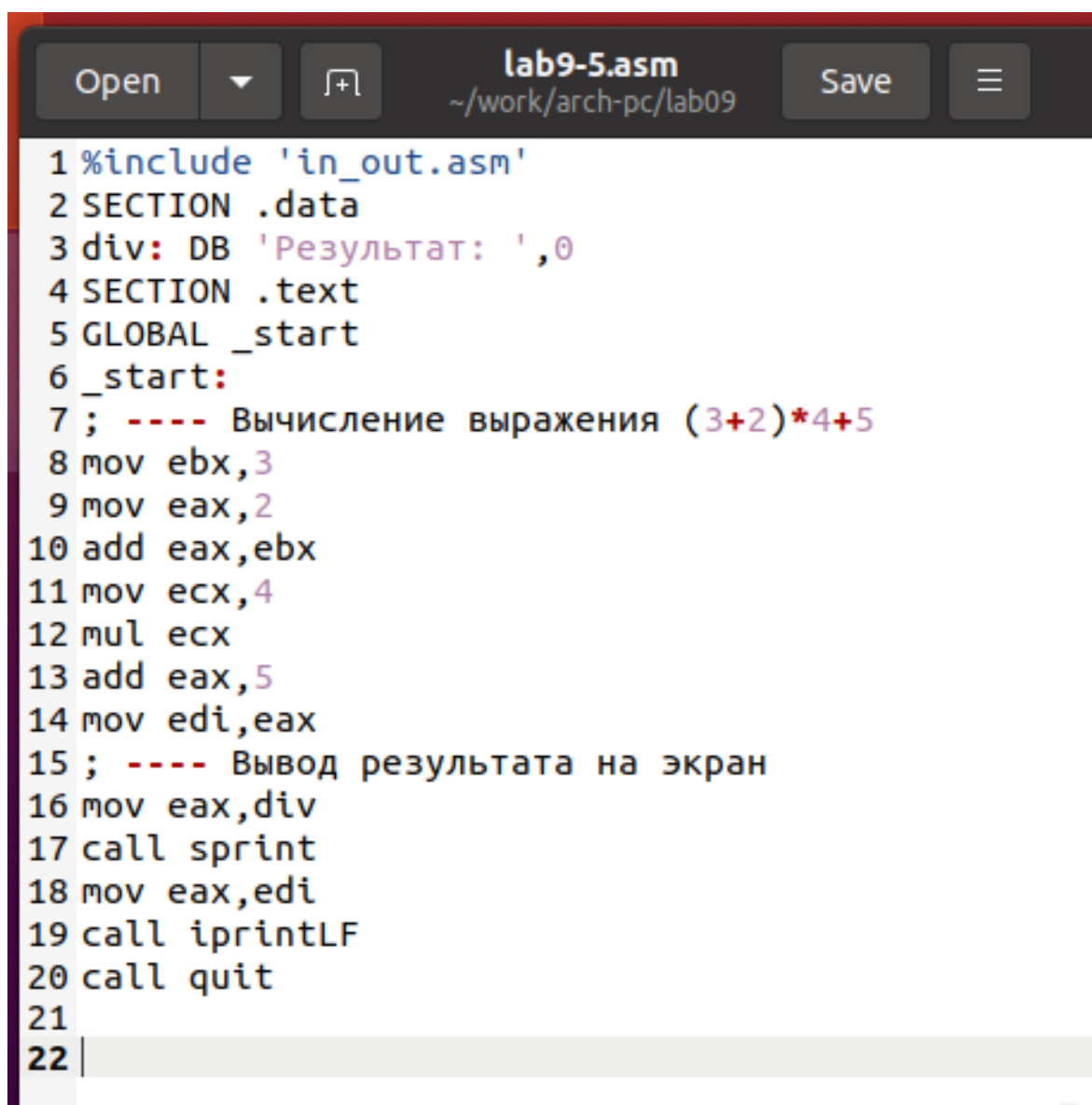
```

```

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

```



```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ---- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add eax,ebx
11 mov ecx,4
12 mul ecx
13 add eax,5
14 mov edi,eax
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
21
22 |
```

Рис. 2.20: код исправлен

```

fgaibullaev@ubuntu: ~/work/arch-pc/lab09
eax      0x5      5
ecx      0x4      4
edx      0x0      0
ebx      0x3      3
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f9 0x80490f9 <_start+17>
eflags   0x206    [ PF IF ]
cs       0x23     35
ss       0x2b     43

B+ 0x80490e8 <_start>      mov     ebx,0x3
B+ 0x80490e8 <_start>5>    mov     ebx,0x3
0x80490ed <_start>5>      mov     eax,0x2
0x80490f2 <_start>10>     add     eax,ebx
>0x80490f4 <_start>12>    mov     ecx,0x4
0x80490f9 <_start>17>    mul     ecx,0x5
0x80490fb <_start>19>     add     eax,0x5
0x80490fe <_start>22>     mov     edi,eax04a000
0x8049100 <_start>24>     mov     eax,0x804a000rint>
0x8049105 <_start>29>     call    0x804900f <sprint>
0x804910a <_start>34>     mov     eax,edi86 <iprintLF>
0x804910c <_start>36>     call    0x8049086 <iprintLF>
0x8049111 <_start>41>     call    0x80490db <quit>

native process 4769 In: _start L?? PC: 0x80490f9
(gdb) rNo process In: me/fgaibullaev/work/arch-pc/la L?? PC: ??
Breakpoint 1, 0x080490e8 in _start ()
(gdb)
(gdb) si0x080490ed in _start ()
(gdb)
(gdb) si0x080490f2 in _start ()
(gdb)
(gdb) si0x080490f4 in _start ()
(gdb) si
0x080490f9 in _start ()
(gdb) cont
Continuing.
Результат: 25
[Inferior 1 (process 4769) exited normally]
(gdb)

```

Рис. 2.21: проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.