

Отчёт по лабораторной работе 9

Архитектура компьютера

Довран Илиев

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	27

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	9
2.5	Программа lab9-2.asm	10
2.6	Запуск программы lab9-2.asm в отладчике	11
2.7	Дизассемблированный код	12
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 lab9-4.asm	21
2.17	Запуск программы lab9-4.asm	22
2.18	Код с ошибкой	23
2.19	Отладка	24
2.20	Код исправлен	25
2.21	Проверка работы	26

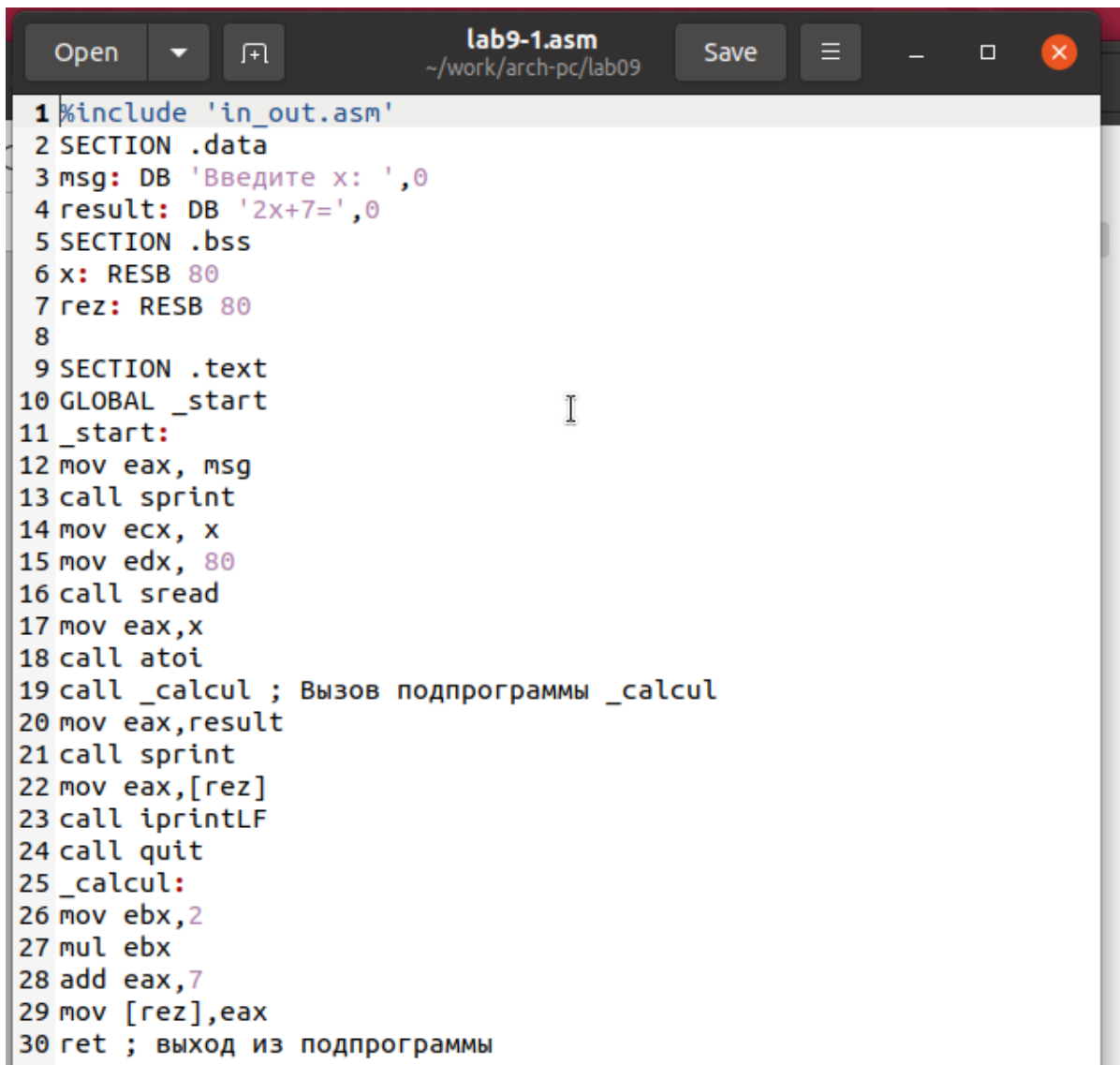
Список таблиц

1 Цель работы

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

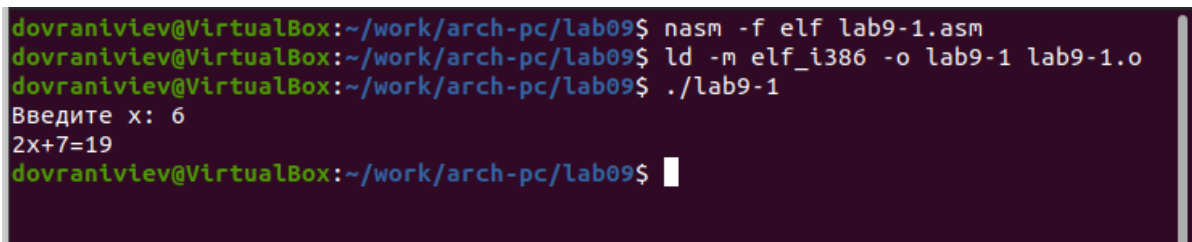
2 Выполнение лабораторной работы

1. Создал директорию для проведения лабораторной работы № 9, перешел в неё и сформировал файл lab9-1.asm.
2. Возьмем для анализа программу, предназначенную для вычисления функции $f(x) = 2x + 7$ с применением вспомогательной функции calcul. В этом случае x получаем через ввод с клавиатуры, а расчет выражения осуществляется внутри вспомогательной функции.



```
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
```

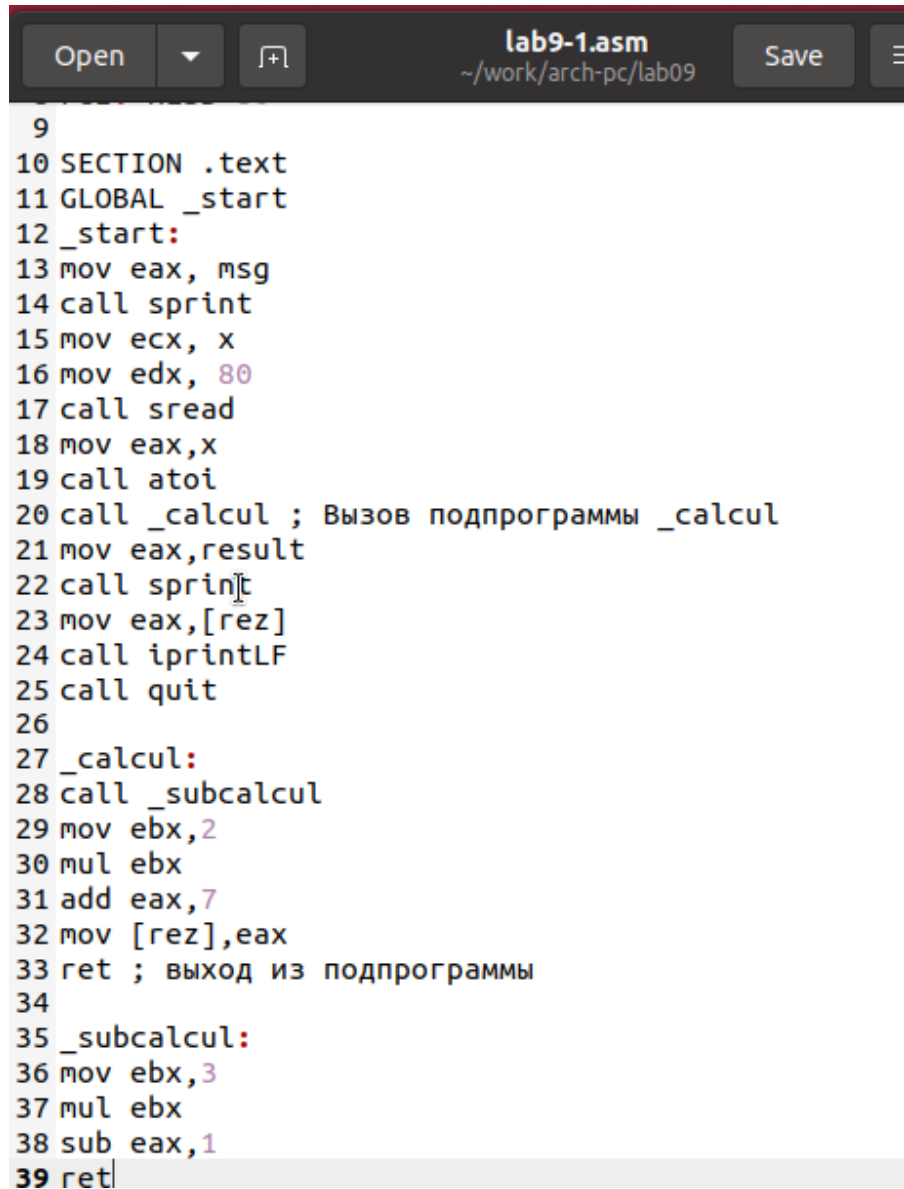
Рис. 2.1: Программа lab9-1.asm



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

Рис. 2.2: Запуск программы lab9-1.asm

3. Изменил код программы, включив вспомогательную функцию `subcalcul` в функцию `calcul` для определения значения составной функции $f(g(x))$, где ввод x происходит с клавиатуры и $f(x) = 2x + 7$, $g(x) = 3x - 1$.



```
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: Программа lab9-1.asm


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

Рис. 2.4: Запуск программы lab9-1.asm

4. Сформировал файл lab9-2.asm, содержащий код программы из Примера 9.2. (Программа выводит на экран сообщение 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: Программа lab9-2.asm

Скомпилировал исходный код в исполняемый файл. Для отладки средствами GDB в исполняемый файл нужно включить отладочную информацию, что достигается компиляцией с опцией `'-g'`.

Запустил исполняемый файл в отладчике gdb. Проверил функционирование программы, выполнив её в среде GDB с помощью команды `run` (или `r`).

```
dovraniviev@VirtualBox:~/work/arch-pc/lab09$  
dovraniviev@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm  
dovraniviev@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o  
dovraniviev@VirtualBox:~/work/arch-pc/lab09$  
dovraniviev@VirtualBox:~/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/dovraniviev/work/arch-pc/lab09/lab9-2  
Hello, world!  
[Inferior 1 (process 3017) 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 дизассемблированный код программы.

```
dovraniviev@VirtualBox: ~/work/arch-pc/lab09
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/dovraniviev/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 3017) exited normally]
(gdb) break _start
Breakpoint 1 at 0x08049000
(gdb) run
Starting program: /home/dovraniviev/work/arch-pc/lab09/lab9-2

Breakpoint 1, 0x08049000 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: Дизассемблированный код

```
dovraniviev@VirtualBox: ~/work/arch-pc/lab09
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) 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

В прошлых этапах была определена точка остановки с именем метки (`_start`). Для проверки использовал команду `info breakpoints` (или сокращенно `i b`). Затем задал дополнительную точку остановки на конкретный адрес инструкции. Чтобы увидеть адрес инструкции, посмотрел на центральную часть экрана, где в левом столбце отображается адрес нужной инструкции. Выбрал адрес для инструкции перед последней (`mov ebx,0x0`) и установил на нем точку остановки.

```
dovraniviev@VirtualBox: ~/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
eip      0x8049000 0x8049000 <_start>

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 3021 In: _start L?? PC: 0x8049000
(gdb)
(gdb)
(gdb) b *0x8049031Breakpoint 2 at 0x8049031
(gdb) i b
Num    Type      Disp Enb Address    What
1      breakpoint keep y  0x08049000 <_start>
       breakpoint already hit 1 time
2      breakpoint keep y  0x08049031 <_start+49>
(gdb) 
```

Рис. 2.9: Точка остановки

Отладчик предоставляет возможность просмотра содержимого памяти и регистров, а также позволяет ручное редактирование значений регистров и переменных. С помощью команды `stepi` (или `si`) выполнил пять инструкций, следя за изменениями в значениях регистров.

```
dovraniviev@VirtualBox: ~/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
eip      0x8049005 0x8049005 <_start+5>

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 3021 In: _start L?? PC: 0x8049005
eflags    0x202    [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
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

```
dovraniviev@VirtualBox: ~/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
eip      0x8049016 0x8049016 <_start+22>

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 3021 In: _start L?? PC: 0x8049016
(gdb) si
0x08049005 in _start ()
(gdb)
(gdb) si0x0804900a in _start ()
(gdb)
(gdb) si0x0804900f in _start ()
(gdb)
(gdb) si0x08049014 in _start ()
(gdb) si
0x08049016 in _start ()
(gdb) 
```

Рис. 2.11: Изменение регистров

Ознакомился со значением переменной `msg1`, используя ее имя. Просмотрел содержимое переменной `msg2` по ее адресу.

Для изменения значений регистров или памяти применил команду `set`, указав в аргументах имя регистра или адрес памяти. Изменил первый символ в переменной `msg1`.

The screenshot shows a GDB debugger window titled "dovraniviev@VirtualBox: ~/work/arch-pc/lab09". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of various registers: `eax` (0x4), `ecx` (0x804a000), `edx` (0x8), `ebx` (0x1), `esp` (0xffffd1d0), `ebp` (0x0), `esi` (0x0), `edi` (0x0), and `eip` (0x804901b). The middle section shows assembly code with addresses and instructions, with the instruction at address 0x804901b (`mov ebx, 0x1`) highlighted. The bottom section shows the GDB prompt with several commands and their outputs, including memory dumps and string displays.

```
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
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 3021 In: _start L?? PC: 0x804901b
(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

Вывел значение регистра `edx` в разных форматах: шестнадцатеричном, двоичном и символьном.

```
dovraniviev@VirtualBox: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
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 3021 In: _start L?? PC: 0x804901b
(gdb)
(gdb) p/s $ecx$3 = 134520832
(gdb)
(gdb) p/x $ecx$4 = 0x804a000
(gdb)
(gdb) p/s $edx$5 = 8
(gdb)
(gdb) p/t $edx$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
```

Рис. 2.13: Вывод значения регистра

Используя команду set, модифицировал значение в регистре ebx.

```
dovraniviev@VirtualBox: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd1d0 0xffffd1d0
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 3021 In: start L?? PC: 0x804901b
(gdb)
(gdb) p/t $edx$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
(gdb)
(gdb) p/s $ebx$8 = 50
(gdb)
(gdb) p/s $ebx
$9 = 2
(gdb)
```

Рис. 2.14: Вывод значения регистра

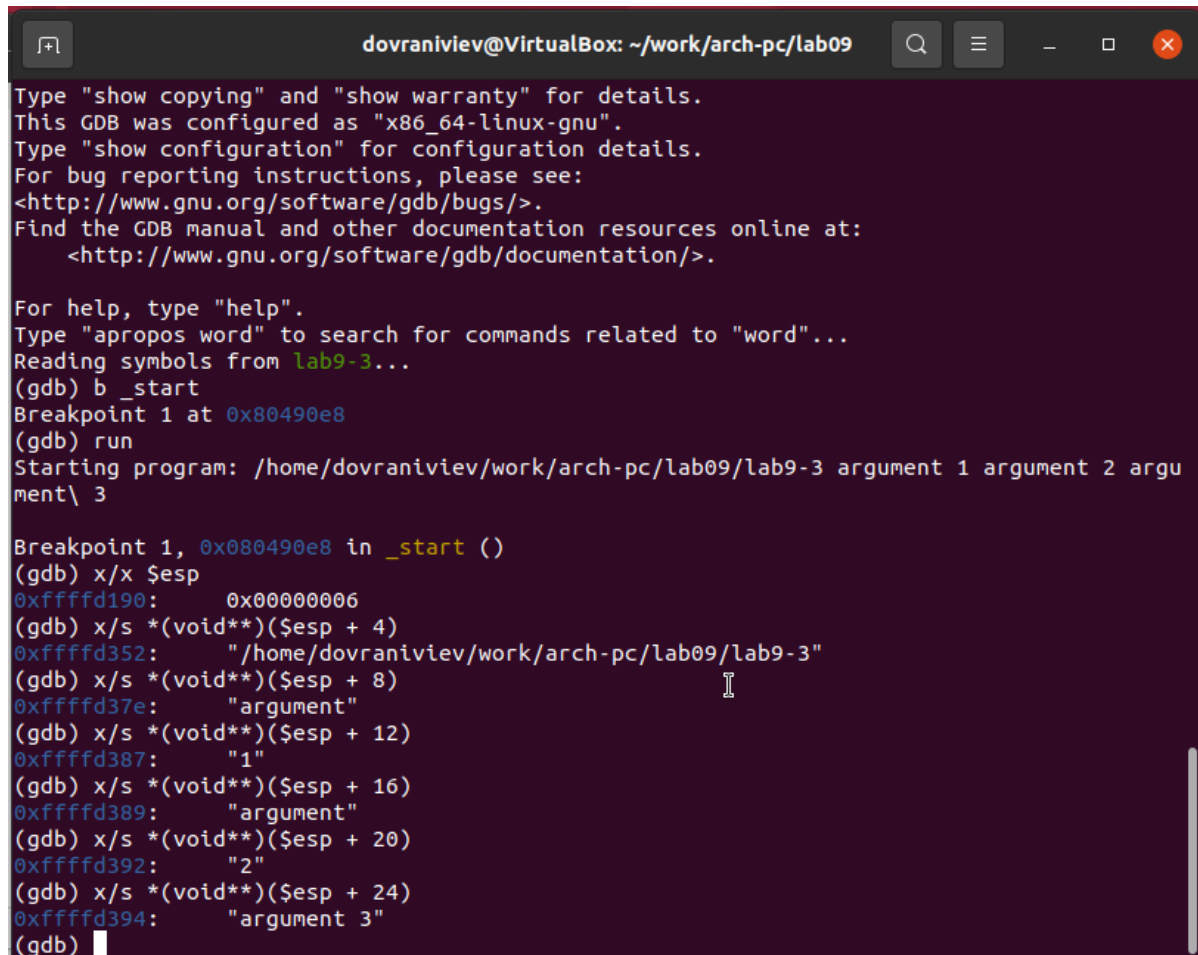
5. Произвел копирование файла lab8-2.asm, который был создан в ходе выполнения лабораторной работы №8 и содержит программу для отображения аргументов командной строки на экране, после чего скомпилировал его в исполняемый файл. Для загрузки исполняемого файла в отладчик gdb с аргументами командной строки применил опцию `-args`. После этого загрузил программу в отладчик, указав необходимые аргументы.

Установил точку прерывания на начало исполнения программы и начал выполнение.

Регистр `esp` содержит адрес вершины стека, где находится количество аргументов командной строки (включая само имя программы). В данном случае об-

наружил, что число аргументов составляет 5, включая имя программы lab9-3 и аргументы: аргумент1, аргумент2 и 'аргумент 3'.

Изучил адреса в стеке: по адресу [esp+4] находится указатель на имя программы, по адресу [esp+8] – на первый аргумент, а по адресу [esp+12] – на второй и так далее.

A screenshot of a GDB terminal window. The window title is 'dovraniviev@VirtualBox: ~/work/arch-pc/lab09'. The terminal shows the GDB startup sequence, including instructions on how to use GDB and where to find documentation. The user sets a breakpoint at the start of the program and runs it. The program is 'lab9-3' with arguments 'argument 1', 'argument 2', and 'argument 3'. The user then inspects the stack pointer register (\$esp) and the stack contents at various offsets. The stack contains the program path, the arguments, and the number 1.

```
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/>.

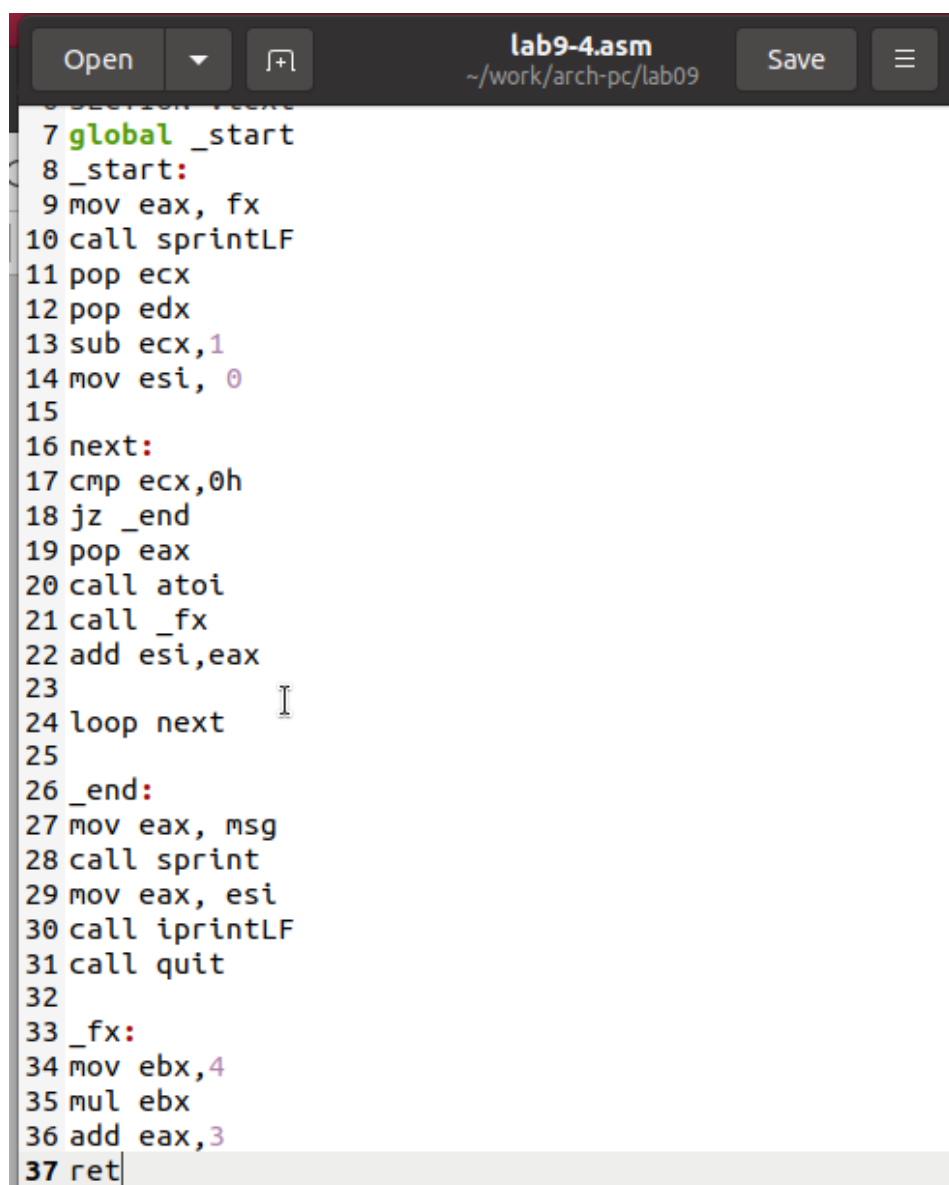
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) run
Starting program: /home/dovraniviev/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argu
ment\ 3

Breakpoint 1, 0x80490e8 in _start ()
(gdb) x/x $esp
0xffffd190: 0x00000006
(gdb) x/s *(void**)($esp + 4)
0xffffd352: "/home/dovraniviev/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd37e: "argument"
(gdb) x/s *(void**)($esp + 12)
0xffffd387: "1"
(gdb) x/s *(void**)($esp + 16)
0xffffd389: "argument"
(gdb) x/s *(void**)($esp + 20)
0xffffd392: "2"
(gdb) x/s *(void**)($esp + 24)
0xffffd394: "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

Размер шага изменения адреса ([esp+4], [esp+8], [esp+12]) составляет 4 байта, что соответствует размеру хранящейся переменной.

6. Модифицировал программу из лабораторной работы №8 (самостоятельное задание №1), внедрив функцию вычисления значения функции $f(x)$ в виде отдельной подпрограммы.



```
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 _fx
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 _fx:
34 mov ebx, 4
35 mul ebx
36 add eax, 3
37 ret
```

Рис. 2.16: Программа lab9-4.asm

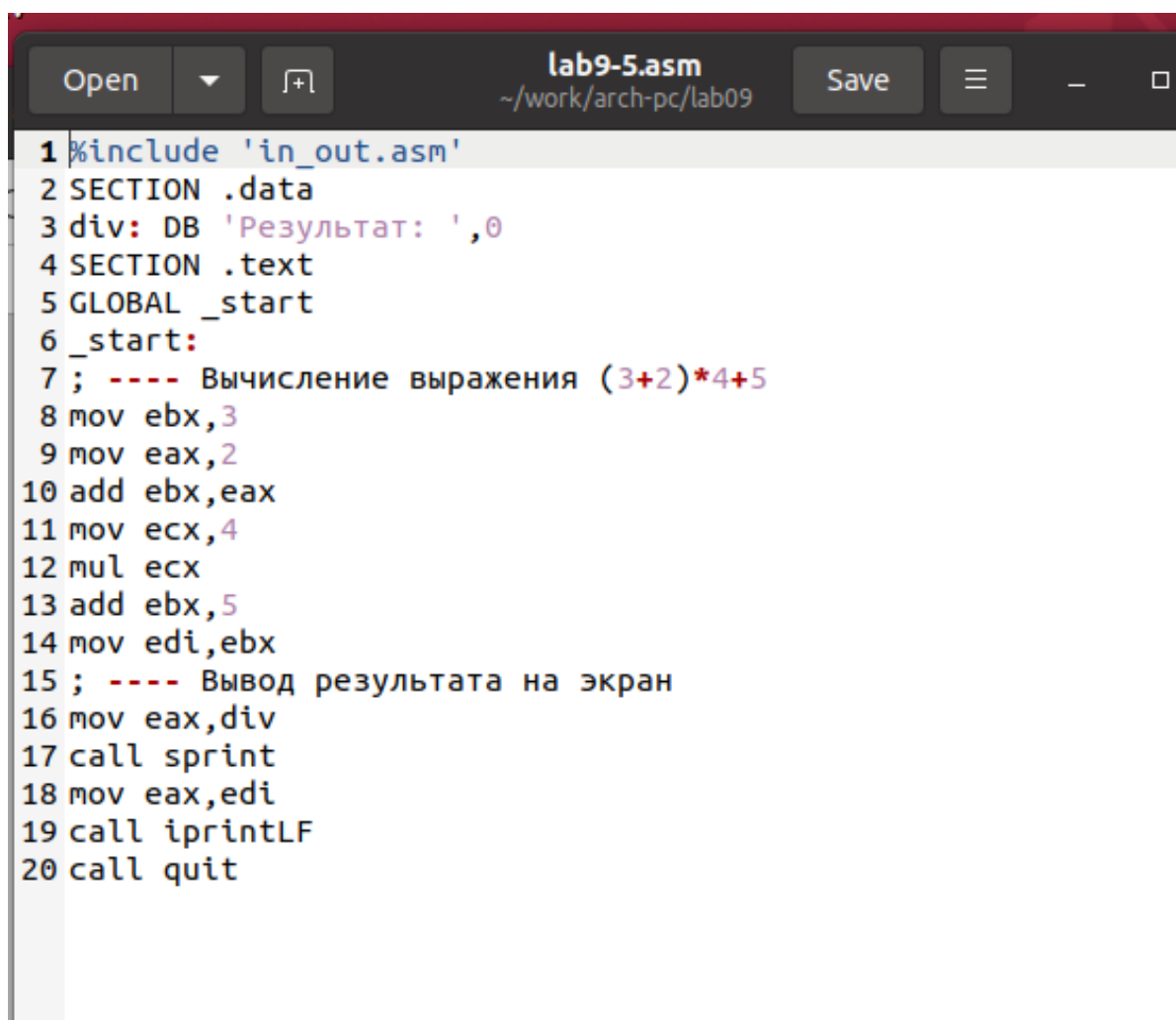
```

dovranivlev@VirtualBox:~/work/arch-pc/lab09$
dovranivlev@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
dovranivlev@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
dovranivlev@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4 1
f(x)= 4x + 3
Результат: 7
dovranivlev@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4 1 3 6 9 7
f(x)= 4x + 3
Результат: 119
dovranivlev@VirtualBox:~/work/arch-pc/lab09$

```

Рис. 2.17: Запуск программы lab9-4.asm

7. В представленном списке команд находится код для расчета выражения $(3 + 2) * 4 + 5$. После его выполнения результат оказался некорректным, что я убедился лично. Используя отладчик GDB и наблюдая за изменениями в регистрах, я выявляю причину ошибки и внесу соответствующие корректировки.



```
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

```
dovraniviev@VirtualBox: ~/work/arch-pc/lab09
eax      0x804a000      134520832
ecx      0x4           4
edx      0x0           0
ebx      0xa           10
esp      0xffffd1d0    0xffffd1d0
ebp      0x0           0x0
esi      0x0           0
edi      0xa           10
eip      0x8049105      0x8049105 <_start+29>

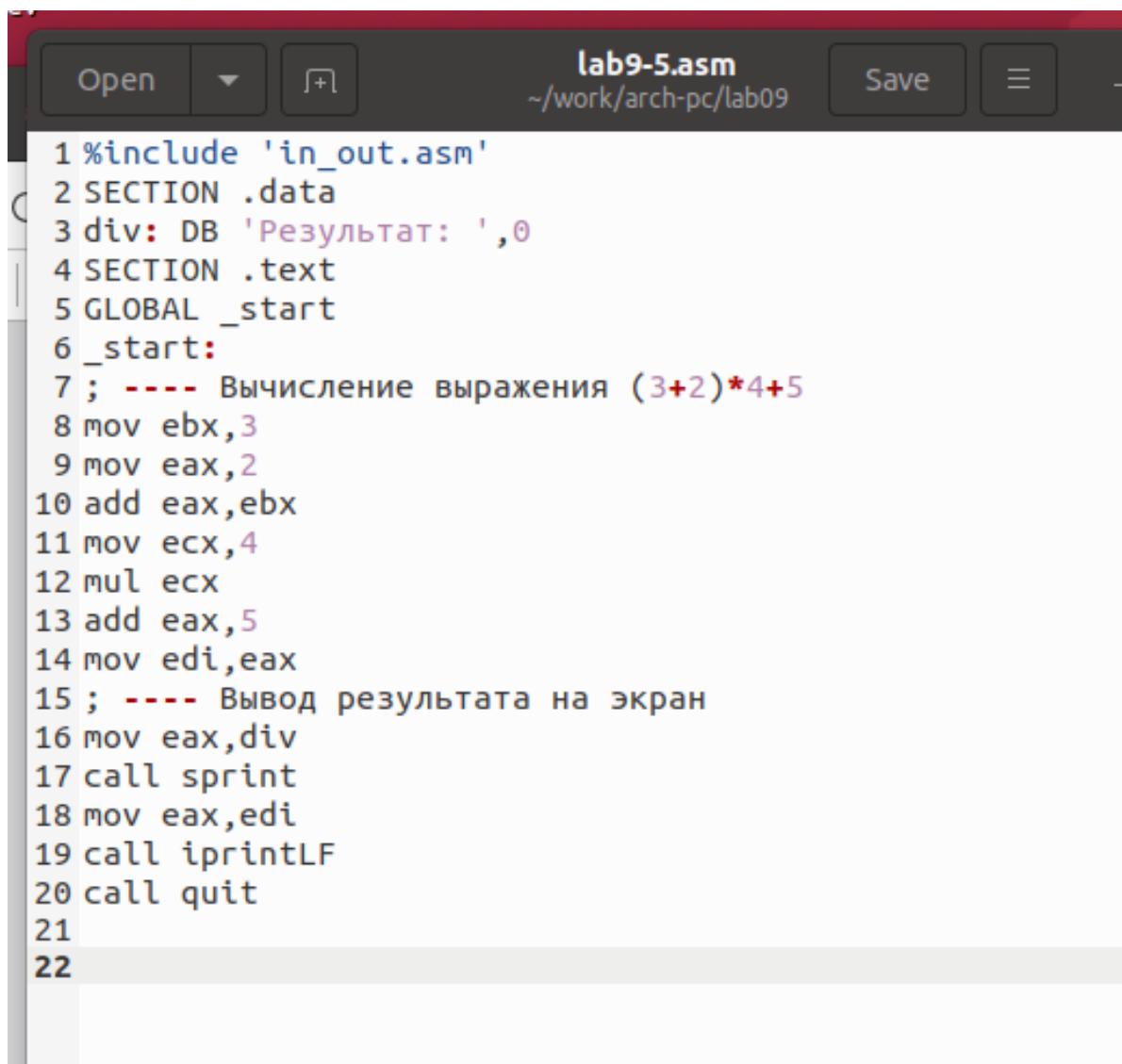
0x80490f9 <_start+17>  mul    ecx
B+ 0x80490e8 <_start>    mov    ebx,0x3
0x80490ed <_start+5>    mov    eax,0x2
0x80490f2 <_start+10>   add    ebx,eax04a000
>0x80490f4 <_start+12>  mov    ecx,0x4      rint>
0x80490f9 <_start+17>  mul    ecx
0x80490fb <_start+19>   add    ebx,0x586 <iprintLF>
0x80490fe <_start+22>  mov    edi,ebx
0x8049100 <_start+24>  mov    eax,0x804a000 x],al
0x8049105 <_start+29>  call   0x804900f <sprint>

native process 3069 In: _start      L??  PC: 0x8049105
(gdb) sNo process In:              L??  PC: ??
(gdb) si
0x080490fe in _start ()
(gdb) si
0x08049100 in _start ()
(gdb) si
0x08049105 in _start ()
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 3069) exited normally]
(gdb) █
```

Рис. 2.19: Отладка

Укажу на то, что был неверно задан порядок операндов в инструкции `add`, а также на то, что в конце выполнения в регистр `edi` записывается значение из `ebx` вместо ожидаемого `eax`.

Вношу изменения в исходный код программы.



```
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

```

dovraniviev@VirtualBox: ~/work/arch-pc/lab09
eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd1d0 0xffffd1d0
ebp      0x0       0x0
esi      0x0       0
edi      0x19      25
eip      0x8049100 0x8049100 <_start+24>

0x80490f9 <_start+17> mul    ecx
B+ 0x80490e8 <_start>    mov    ebx,0x3
0x80490ed <_start+5>    mov    eax,0x2
>0x80490f2 <_start+10>   add    eax,ebx04a000
0x80490f4 <_start+12>   mov    ecx,0x4    rint>
0x80490f9 <_start+17>   mul    ecx
0x80490fb <_start+19>   add    eax,0x586 <iprintLF>
0x80490fe <_start+22>   mov    edi,eax
0x8049100 <_start+24>   mov    eax,0x804a000 x],al
0x8049105 <_start+29>   call   0x804900f <sprint>

native process 3081 In: _start L?? PC: 0x8049100
(gdb) sNo process In: L?? PC: ??
(gdb) si
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) si
0x08049100 in _start ()
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 3081) exited normally]
(gdb)

```

Рис. 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.