

Лабораторная работа №9

Понятие подпрограммы. Отладчик.

Турсунбоев Сардорбек

Содержание

| | | |
|----------|---------------------------------------|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Выполнение лабораторной работы | 7 |
| 4 | Выводы | 27 |

Список иллюстраций

| | | |
|------|---|----|
| 3.1 | Код программы lab9-1.asm | 8 |
| 3.2 | Тестирование программы lab9-1.asm | 8 |
| 3.3 | Код программы lab9-1.asm | 9 |
| 3.4 | Тестирование программы lab9-1.asm | 10 |
| 3.5 | Код программы lab9-2.asm | 11 |
| 3.6 | Тестирование программы lab9-2.asm в отладчике | 12 |
| 3.7 | Дизассемблированный код | 13 |
| 3.8 | Дизассемблированный код в режиме интел | 13 |
| 3.9 | Точка остановки | 14 |
| 3.10 | Изменение регистров | 15 |
| 3.11 | Изменение регистров | 16 |
| 3.12 | Изменение значения переменной | 17 |
| 3.13 | Вывод значения регистра | 18 |
| 3.14 | Вывод значения регистра | 19 |
| 3.15 | Вывод значения регистра | 20 |
| 3.16 | Код программы lab9-4.asm | 21 |
| 3.17 | Тестирование программы lab9-4.asm | 22 |
| 3.18 | Код с ошибкой | 23 |
| 3.19 | Отладка | 24 |
| 3.20 | Код исправлен | 25 |
| 3.21 | 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.

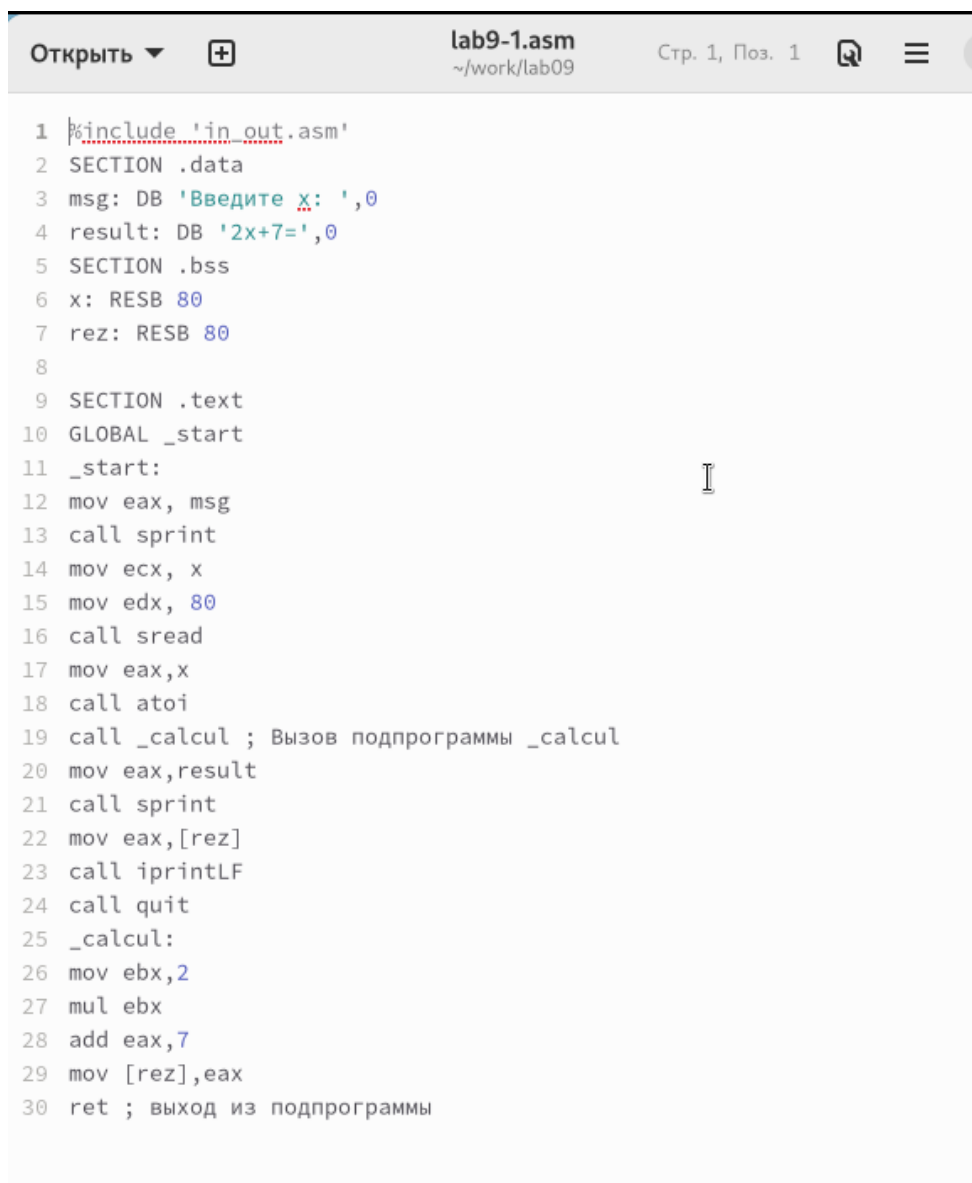
2 Задание

1. Изучить использование подпрограмм
2. Изучить работу с отладчиком
3. Рассмотреть примеры программ
4. Рассмотреть пример передачи аргументов
5. Выполнить самостоятельное задание
6. Найти ошибку в программе

3 Выполнение лабораторной работы

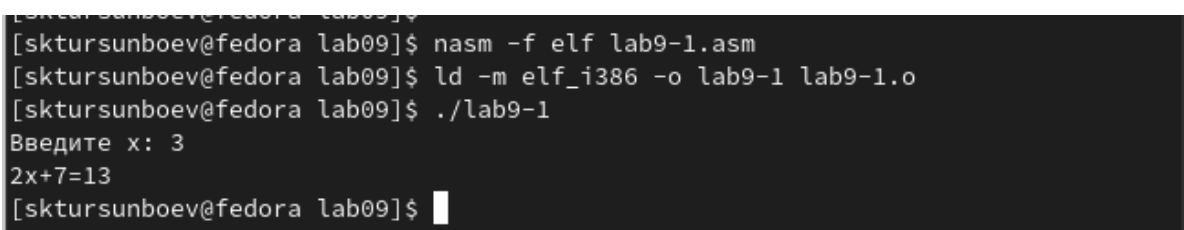
Я создал каталог для выполнения лабораторной работы № 9 и перешел в него. Затем я создал файл lab9-1.asm.

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.



```
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
```

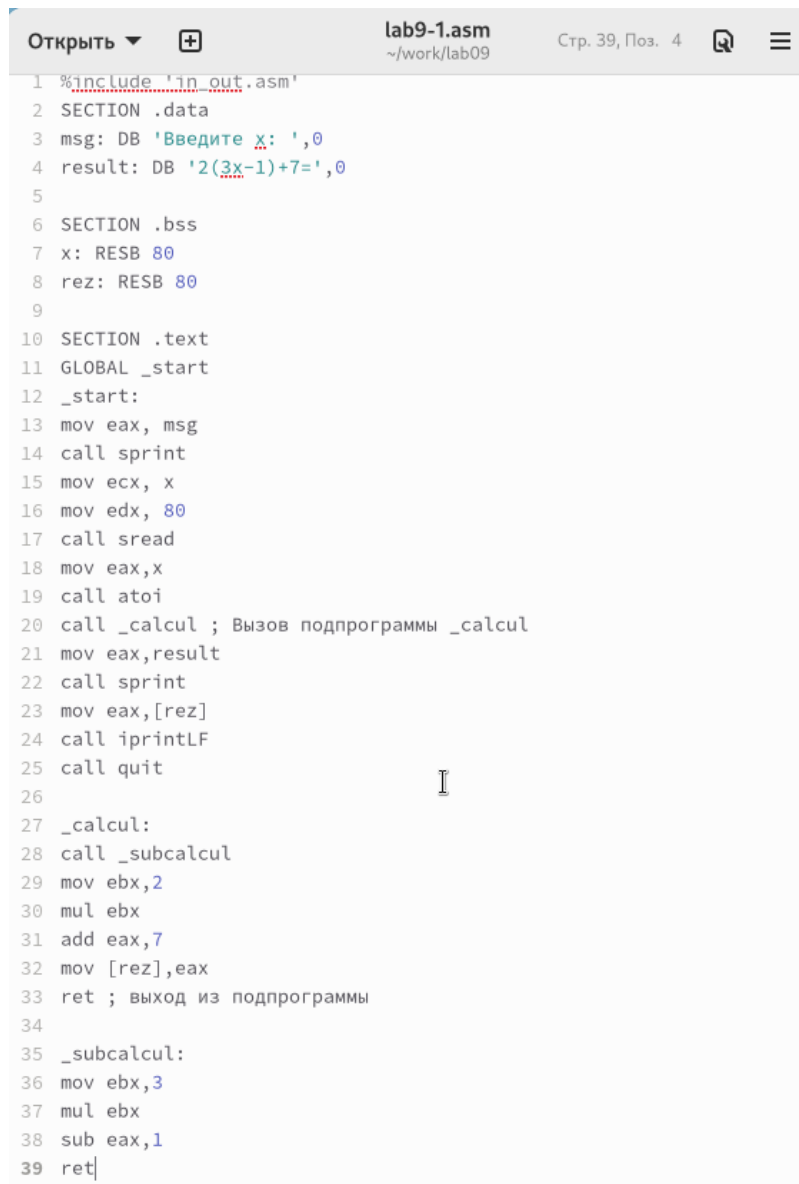
Рис. 3.1: Код программы lab9-1.asm



```
[sktursunboev@fedora lab09]$ nasm -f elf lab9-1.asm
[sktursunboev@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[sktursunboev@fedora lab09]$ ./lab9-1
Введите x: 3
2x+7=13
[sktursunboev@fedora lab09]$
```

Рис. 3.2: Тестирование программы lab9-1.asm

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



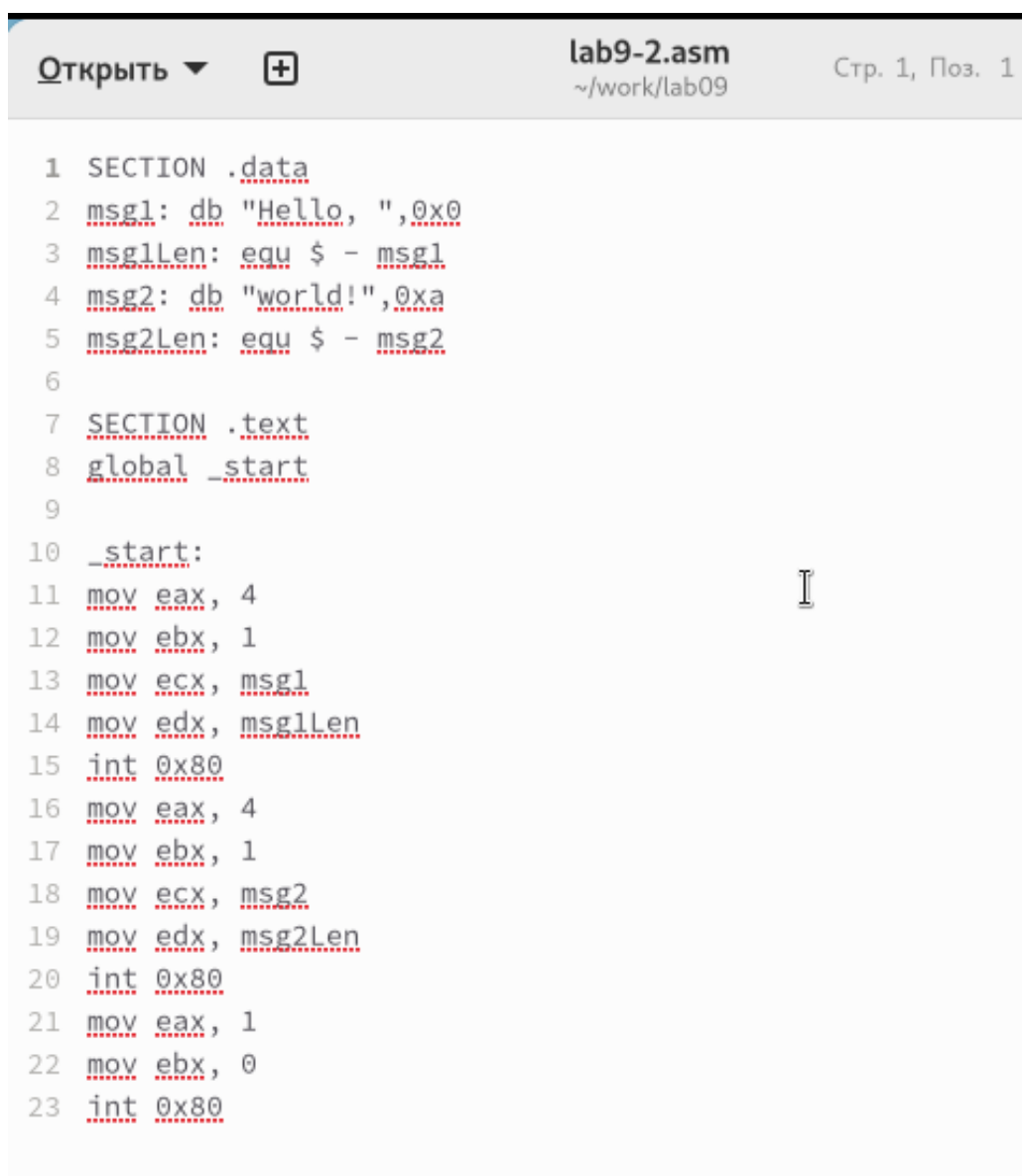
```
Открыть ▾ + lab9-1.asm Стр. 39, Поз. 4
~/work/lab09
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2(3x-1)+7=',0
5
6 SECTION .bss
7 x: RESB 80
8 rez: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
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. 3.3: Код программы lab9-1.asm

```
[sktursunboev@fedora lab09]$ nasm -f elf lab9-1.asm
[sktursunboev@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[sktursunboev@fedora lab09]$ ./lab9-1
Введите x: 3
2x+7=13
[sktursunboev@fedora lab09]$ nasm -f elf lab9-1.asm
[sktursunboev@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[sktursunboev@fedora lab09]$ ./lab9-1
Введите x: 3
2(3x-1)+7=23
[sktursunboev@fedora lab09]$
```

Рис. 3.4: Тестирование программы lab9-1.asm

Создал 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. 3.5: Код программы 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').

```

[sktursunboev@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[sktursunboev@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[sktursunboev@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/sktursunboev/work/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 6380) exited normally]

```

Рис. 3.6: Тестирование 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.

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/sktursunboev/work/lab09/lab9-2
```

```
Breakpoint 1, _start () at lab9-2.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. 3.7: 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
```

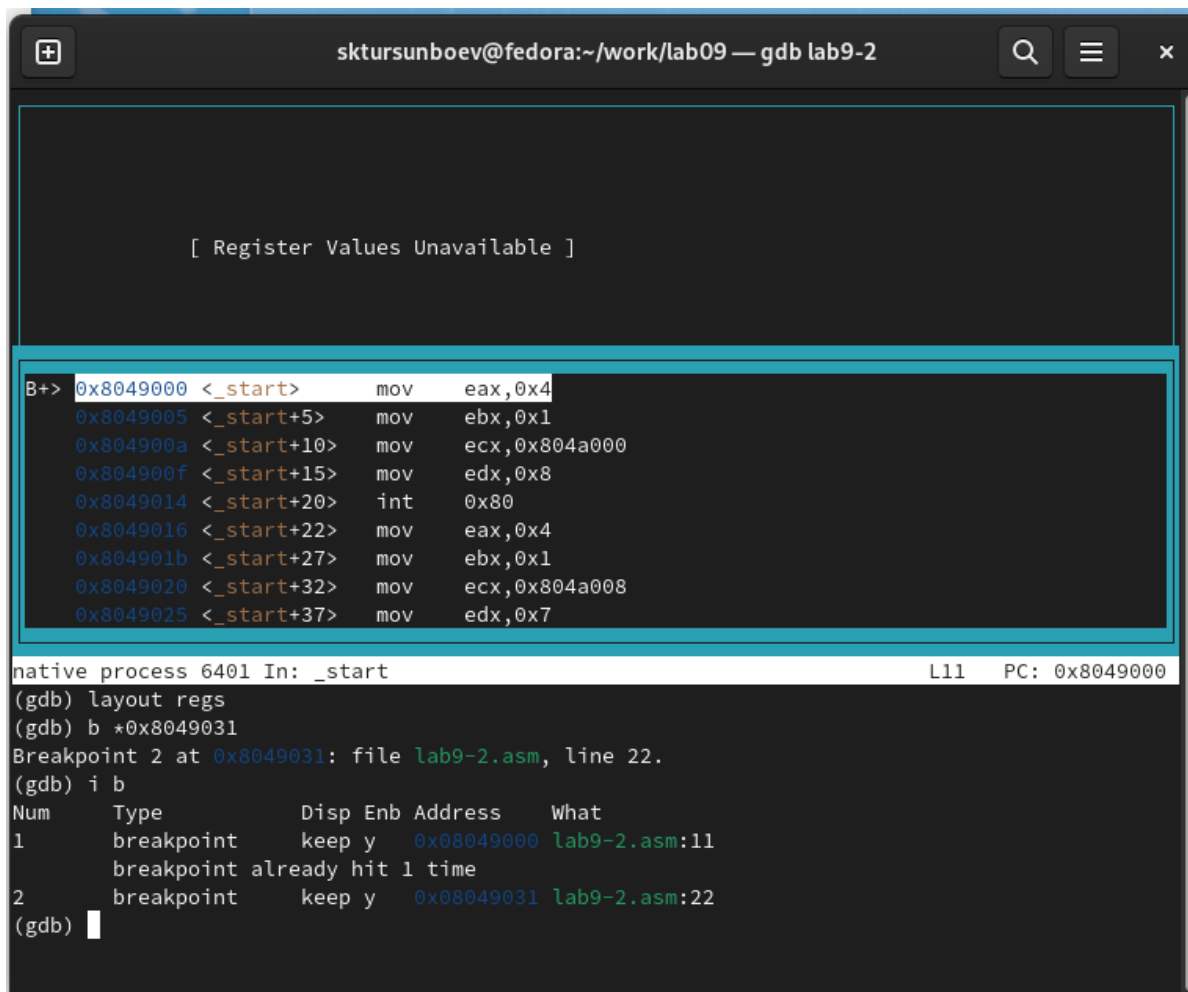
End of assembler dump.

```
(gdb) █
```

Рис. 3.8: 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’.



The screenshot shows the GDB interface with the following content:

```
sktursunboev@fedora:~/work/lab09 — gdb lab9-2
```

[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
0x8049020 <_start+32>     mov    ecx,0x804a008
0x8049025 <_start+37>     mov    edx,0x7
```

```
native process 6401 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. 3.9: 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.

```
sktursunboev@fedora:~/work/lab09 — gdb lab9-2

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd210 0xffffd210
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
0x8049025 <_start+37> mov edx,0x7

native process 6401 In: _start L12 PC: 0x8049005
eip      0x8049000 0x8049000 <_start>
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
(gdb) 
```

Рис. 3.10: Изменение регистров

```
sktursunboev@fedora:~/work/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd210 0xffffd210
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
    0x8049025 <_start+37> mov    edx,0x7

native process 6401 In: _start L16 PC: 0x8049016
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) |
```

Рис. 3.11: Изменение регистров

Просмотрел значение переменной msg1 по имени и получил нужные данные.
Просмотрел значение переменной msg1 по имени и получил нужные данные.
Для изменения значения регистра или ячейки памяти использовал команду set, указав имя регистра или адрес в качестве аргумента. Изменил первый символ переменной msg1.


```
sktursunboev@fedora:~/work/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd210 0xffffd210
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
    0x8049025 <_start+37>  mov     edx,0x7

native process 6401 In: _start      L16  PC: 0x8049016
(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>:      "Lor!d!\n\034"
(gdb) 
```

Рис. 3.12: 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`.

```
sktursunboev@fedora:~/work/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd210 0xffffd210
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
    0x8049025 <_start+37> mov    edx,0x7

native process 6401 In: _start L16 PC: 0x8049016
(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. 3.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx на нужное значение.

```
sktursunboev@fedora:~/work/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd210 0xffffd210
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
   0x8049025 <_start+37> mov    edx,0x7

native process 6401 In: _start L16 PC: 0x8049016
(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. 3.14: 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.

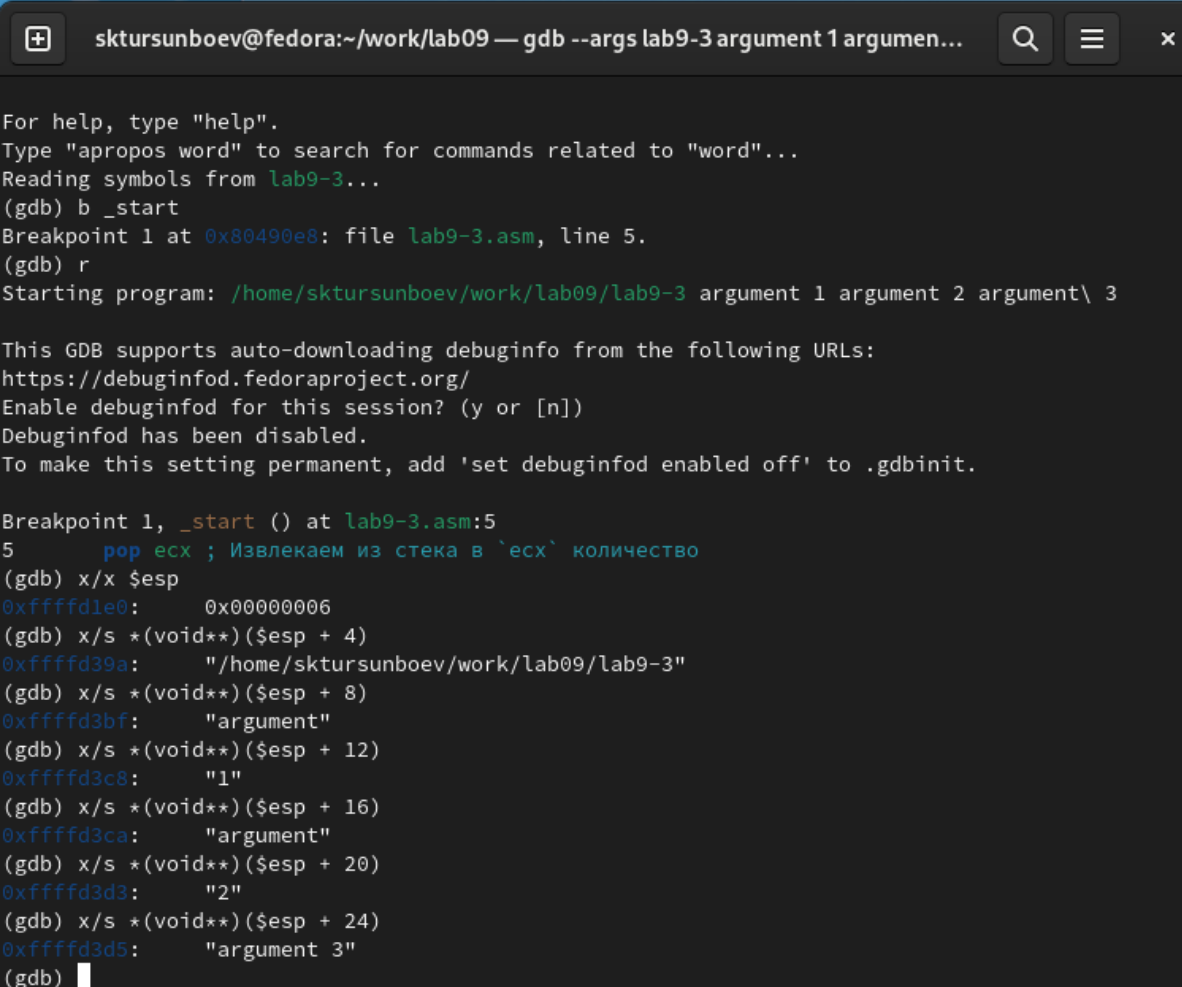
Для загрузки программы с аргументами в gdb использовал ключ `-args` и загрузил исполняемый файл в отладчик с указанными аргументами.

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.



```
sktursunboev@fedora:~/work/lab09 — gdb --args lab9-3 argument 1 argumen...
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) r
Starting program: /home/sktursunboev/work/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.

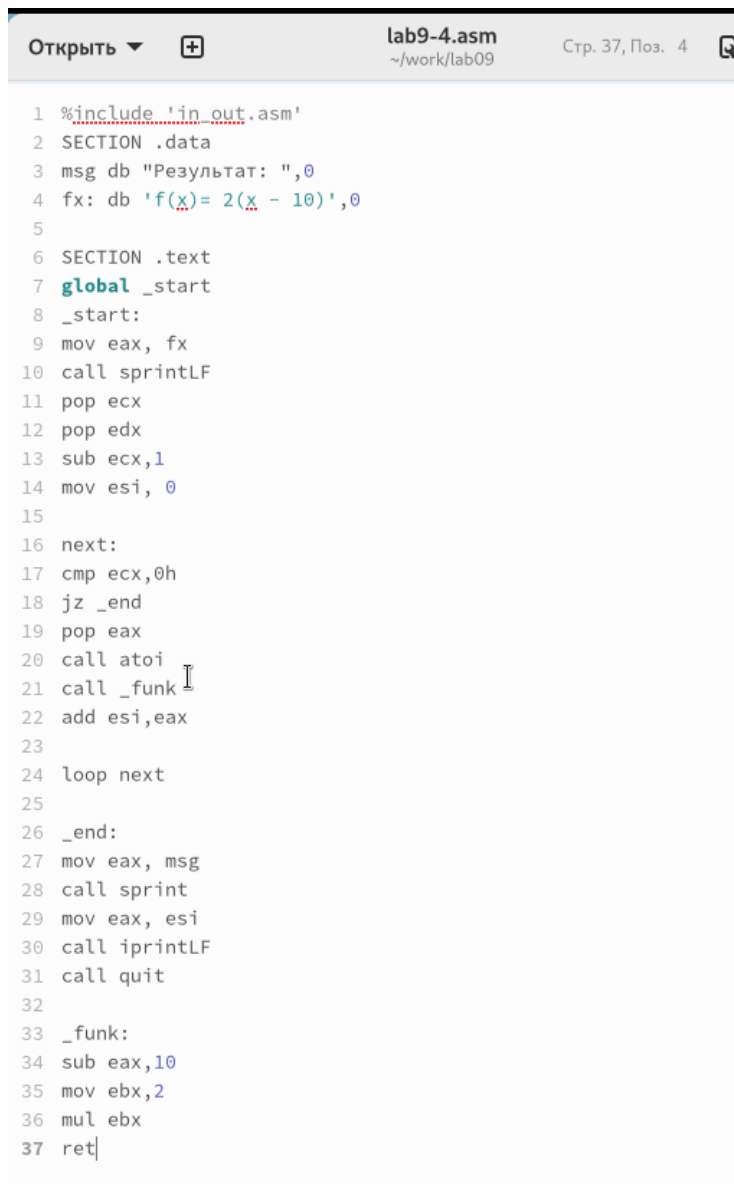
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1e0:      0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd39a:      "/home/sktursunboev/work/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3bf:      "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3c8:      "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3ca:      "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3d3:      "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3d5:      "argument 3"
(gdb)
```

Рис. 3.15: 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]).

Преобразовал программу из лабораторной работы №8 (Задание №1 для само-

стоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.



```
Открыть ▼ + lab9-4.asm ~./work/lab09 Стр. 37, Поз. 4
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 2(x - 10)',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 _funk
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 _funk:
34 sub eax,10
35 mov ebx,2
36 mul ebx
37 ret
```

Рис. 3.16: Код программы lab9-4.asm

```


[sktursunboev@fedora lab09]$
[sktursunboev@fedora lab09]$ nasm -f elf lab9-4.asm
[sktursunboev@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o
[sktursunboev@fedora lab09]$ ./lab9-4
f(x)= 2(x - 10)
Результат: 0
[sktursunboev@fedora lab09]$ ./lab9-4 10
f(x)= 2(x - 10)
Результат: 0
[sktursunboev@fedora lab09]$ ./lab9-4 11
f(x)= 2(x - 10)
Результат: 2
[sktursunboev@fedora lab09]$ ./lab9-4 11 12 13 14 15
f(x)= 2(x - 10)
Результат: 30
[sktursunboev@fedora lab09]$

```

Рис. 3.17: Тестирование программы lab9-4.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. [3.18])

Открыть ▾ 

lab9-5.asm
~/work/lab09



Стр. 1, Поз. 1  

Рис. 3.18: Код с ошибкой

The screenshot shows a GDB window titled "sktursunboev@fedora:~/work/lab09 — gdb lab9-5". The register window at the top lists: eax (0x8, 8), ecx (0x4, 4), edx (0x0, 0), ebx (0xa, 10), esp (0xffffd210, 0xffffd210), ebp (0x0, 0x0), esi (0x0, 0), and edi (0xa, 10). Below this, a list of assembly instructions is shown with addresses: 0x80490f4 <_start+12> mov ecx,0x4; 0x8049100 <_start+24> mov eax,0x804a000rint>; 0x804910a <_start+34> mov eax,edi; 0x804910c <_start+36> call 0x8049086 <iprintLF>; 0x8049111 <_start+41> call 0x80490db <quit>. The bottom panel shows the GDB prompt with commands: (gdb) si, (gdb) si, (gdb) si, (gdb) si, (gdb) si, (gdb) si, (gdb) cont, Continuing., and the output: Результат: 10, [Inferior 1 (process 6686) exited normally]. The status bar at the bottom indicates "native process 6686 In: _start" and "Breakpoint 1: 9-5.asm:8".

```
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa     10
esp      0xffffd210 0xffffd210
ebp      0x0      0x0
esi      0x0      0
edi      0xa     10

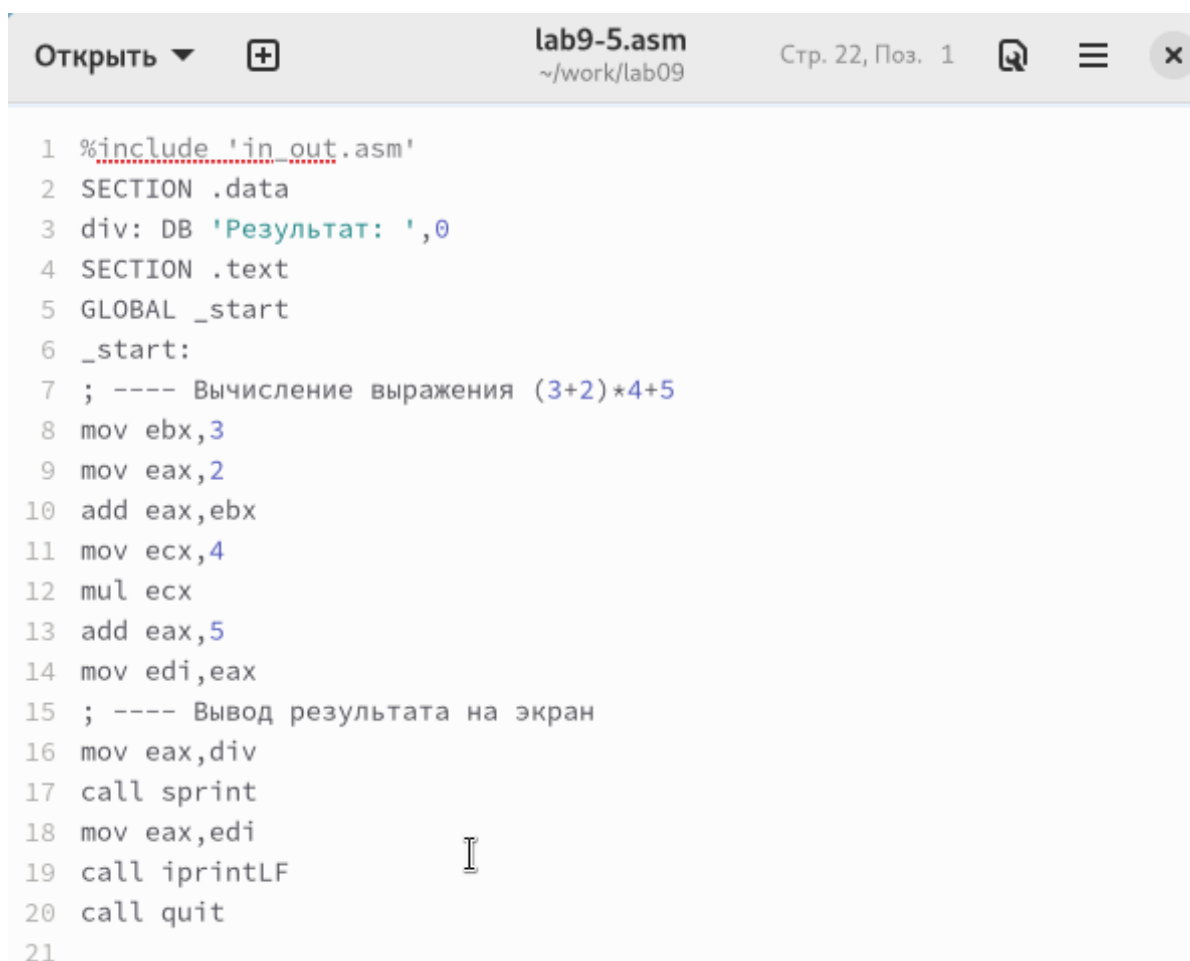
0x80490f4 <_start+12> mov    ecx,0x4
0x8049100 <_start+24> mov    eax,0x804a000rint>
0x804910a <_start+34> mov    eax,edi
0x804910c <_start+36> call   0x8049086 <iprintLF>
0x8049111 <_start+41> call   0x80490db <quit>

native process 6686 In: _start L16 PC: 0x8049100
Breakpoint 1: 9-5.asm:8 L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 10
[Inferior 1 (process 6686) exited normally]
(gdb)
```

Рис. 3.19: Отладка

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

Исправленный 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
```

Рис. 3.20: Код исправлен

```
sktursunboev@fedora:~/work/lab09 — gdb lab9-5

eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd210 0xffffd210
ebp      0x0       0
esi      0x0       0
edi      0x19      25

0x80490f4 <_start+12> mov    ecx,0x4
0x8049100 <_start+24> mov    eax,0x804a000<rint>
0x804910a <_start+34> mov    eax,edi
0x804910c <_start+36> call   0x8049086 <iprintLF>
0x8049111 <_start+41> call   0x80490db <quit>

native process 6742 In: _start L16 PC: 0x8049100
Breakpoint 0: No process In: 0-5.asm:8 L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 25
[Inferior 1 (process 6742) exited normally]
(gdb)
```

Рис. 3.21: Проверка работы

4 Выводы

Освоили работу с подпрограммами и отладчиком.