

Отчет по лабораторной работе №9

Дисциплина: Архитектура компьютерных наук

Литвинов Максим Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программ с помощью GDB	10
2.3	Задание для самостоятельной работы	22
3	Выводы	29

Список иллюстраций

2.1	Изменение кода	7
2.2	Запуск программы	8
2.3	Изменение кода	9
2.4	Запуск программы	10
2.5	Изменение кода	11
2.6	Запуск программы в отладчике	12
2.7	Дизассимилированный код	13
2.8	Дизассимилированный код в режиме интел	14
2.9	Точка остановки	15
2.10	Изменение регистров	16
2.11	Изменение регистров	17
2.12	Изменение значения переменной	18
2.13	Вывод значения регистра	19
2.14	Вывод значения регистра	20
2.15	Изменение кода	21
2.16	Вывод значения регистра	22
2.17	Изменение кода	23
2.18	Запуск программы	24
2.19	Код с ошибкой	25
2.20	Отладка	26
2.21	Код исправлен	27
2.22	Проверка работы	28

Список таблиц

1 Цель работы

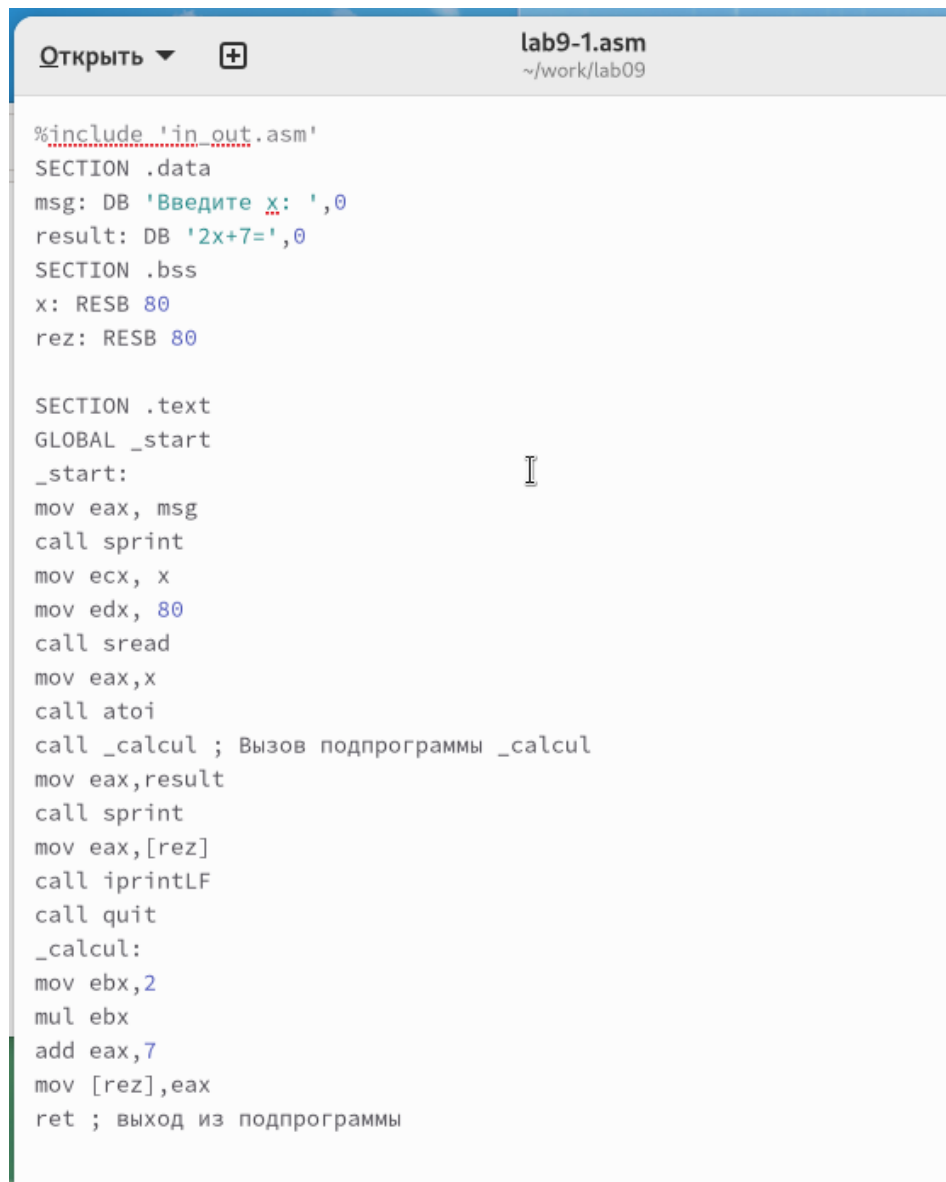
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

Я создал каталог, предназначенный для выполнения лабораторной работы №9, и перешел в него.

В рамках примера рассмотрим программу, которая вычисляет арифметическое выражение $f(x) = 2x + 7$ с использованием подпрограммы `calcul`. В данном примере значение переменной x вводится с клавиатуры, а само выражение вычисляется внутри подпрограммы.




```
Открыть ▾  lab9-1.asm  
~/work/lab09  
  
%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: Изменение кода

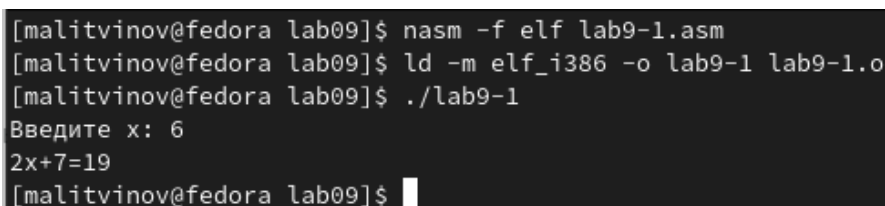
Первые строки программы отвечают за вывод сообщения на экран с использованием функции `sprint`, чтение данных, введенных с клавиатуры с помощью функции `sread`, и преобразование введенных данных из символьного в числовой формат с помощью функции `atoi`.

После инструкции `call _calcul`, которая передает управление подпрограмме `_calcul`, выполнение программы переходит к инструкциям, содержащимся внутри

подпрограммы.

Инструкция `ret` является последней в подпрограмме и ее выполнение приводит к возврату в основную программу к инструкции, следующей за инструкцией `call`, которая вызвала данную подпрограмму.

Внесены изменения в текст программы, добавлена подпрограмма `subcalcul` внутри подпрограммы `calcul` для вычисления выражения $f(g(x))$, где значение x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$.



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

Рис. 2.2: Запуск программы

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
~/work/lab09

msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0

SECTION .bss
x: RESB 80
rez: RESB 80

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

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

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

Рис. 2.3: Изменение кода

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

Рис. 2.4: Запуск программы

2.2 Отладка программ с помощью GDB

Я создал файл с именем lab9-2.asm, в котором содержится текст программы из Листинга 9.2, реализующей функцию печати сообщения “Hello world!”.



```
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: Изменение кода

После компиляции получил исполняемый файл. Чтобы использовать отладчик GDB, я добавил отладочную информацию к исполняемому файлу, указав ключ “-g” при компиляции.

Затем я загрузил исполняемый файл в отладчик GDB и проверил его работу, запустив программу с помощью команды “run” (или “r” в сокращенной форме).

```

[malitvinov@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[malitvinov@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[malitvinov@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) r
Starting program: /home/malitvinov/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 2753) exited normally]
(gdb) █

```

Рис. 2.6: Запуск программы в отладчике

Для более детального анализа программы я установил точку остановки на метке “start”, с которой начинается выполнение любой ассемблерной программы, и запустил ее. Затем я просмотрел дизассемблированный код программы.

```

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

Breakpoint 1, _start () at lab9-2.asm:11
11      mov eax, 4
(gdb)
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(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     $0x1,%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: Дизассимилированный код

```

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: Дизассимилированный код в режиме интел

Чтобы установить точку остановки, я использовал команду “break” (или “b” в сокращенной форме). Типичным аргументом для этой команды может быть номер строки программы, имя метки или адрес. Чтобы избежать путаницы с номерами, перед адресом ставится знак “*“.

На предыдущих шагах я уже установил точку остановки по имени метки “_start” и проверил это с помощью команды “info breakpoints” (или “i b” в сокращенной форме). Затем я установил еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции “mov ebx, 0x0”.

The screenshot shows a GDB terminal window with the title bar "malitvinov@fedora:~/work/lab09 — gdb lab9-2". The main window displays "[Register Values Unavailable]". Below this, a list of assembly instructions is shown, starting with "B+> 0x8049000 <_start> mov eax,0x4". The instructions are as follows:

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

Below the assembly list, the status bar shows "native process 2757 In: _start L11 PC: 0x8049000". The terminal output shows the following commands and responses:

```
(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

Я использовал отладчик, который позволяет просматривать содержимое ячеек памяти и регистров, а также вносить в них изменения при необходимости. Я выполнил 5 инструкций с помощью команды `stepi` (или `si`) и следил за изменениями значений регистров.

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

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd230 0xffffd230
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 2757 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. 2.10: Изменение регистров

The screenshot shows a GDB terminal window with the title bar "malitvinov@fedora:~/work/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the current values of the general-purpose registers:

Register	Value (hex)	Value (dec)
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd230	0xffffd230
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0

. The middle section displays assembly code for the current instruction set, with the instruction at address 0x8049016 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
```

. The bottom section shows the state of the native process 2757, including segment registers (es: 0x2b, fs: 0x0, gs: 0x0) and the current instruction pointer (PC: 0x8049016). The GDB prompt shows several "si" (step instruction) commands and an "is" (step function) command that results in an "Undefined command" error.

Рис. 2.11: Изменение регистров

Далее я просмотрел значение переменной `msg1`, обратившись к ней по имени. Также я просмотрел значение переменной `msg2`, обратившись к ней по адресу.

Для изменения значения регистра или ячейки памяти я использовал команду `set`, указав в качестве аргумента имя регистра или адрес. Я изменил первый символ переменной `msg1`.

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

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd230 0xffffd230
ebp      0x0      0
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 2757 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>: "Lorld!\n\034"
(gdb)
```

Рис. 2.12: Изменение значения переменной

Для вывода значения регистра `edx` в различных форматах (шестнадцатеричном, двоичном и символьном) я использовал соответствующие команды.

The screenshot shows a GDB terminal window with the title bar "malitvinov@fedora:~/work/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of general-purpose registers:

eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd230	0xffffd230
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0

. The middle section shows assembly code with addresses and instructions:

```
B+ 0x8049000 <_start>    mov    eax,0x4
0x8049005 <_start+5>    mov    ebx,0x1
0x804900a <_start+10>   mov    ecx,0x804a000
0x804900f <_start+15>   mov    edx,0x8
0x8049014 <_start+20>   int     0x80
> 0x8049016 <_start+22>   mov    eax,0x4
0x804901b <_start+27>   mov    ebx,0x1
0x8049020 <_start+32>   mov    ecx,0x804a008
0x8049025 <_start+37>   mov    edx,0x7
```

. The bottom section shows the GDB prompt and a series of commands and their outputs:

```
native process 2757 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. 2.13: Вывод значения регистра

С помощью команды `set` также изменил значение регистра `ebx`.

The screenshot shows a GDB terminal window titled "malitvinov@fedora:~/work/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of general-purpose registers:

Register	Value (hex)	Value (decimal)
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x2	2
esp	0xffffd230	0xffffd230
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0

. The middle section displays assembly code with addresses and instructions:

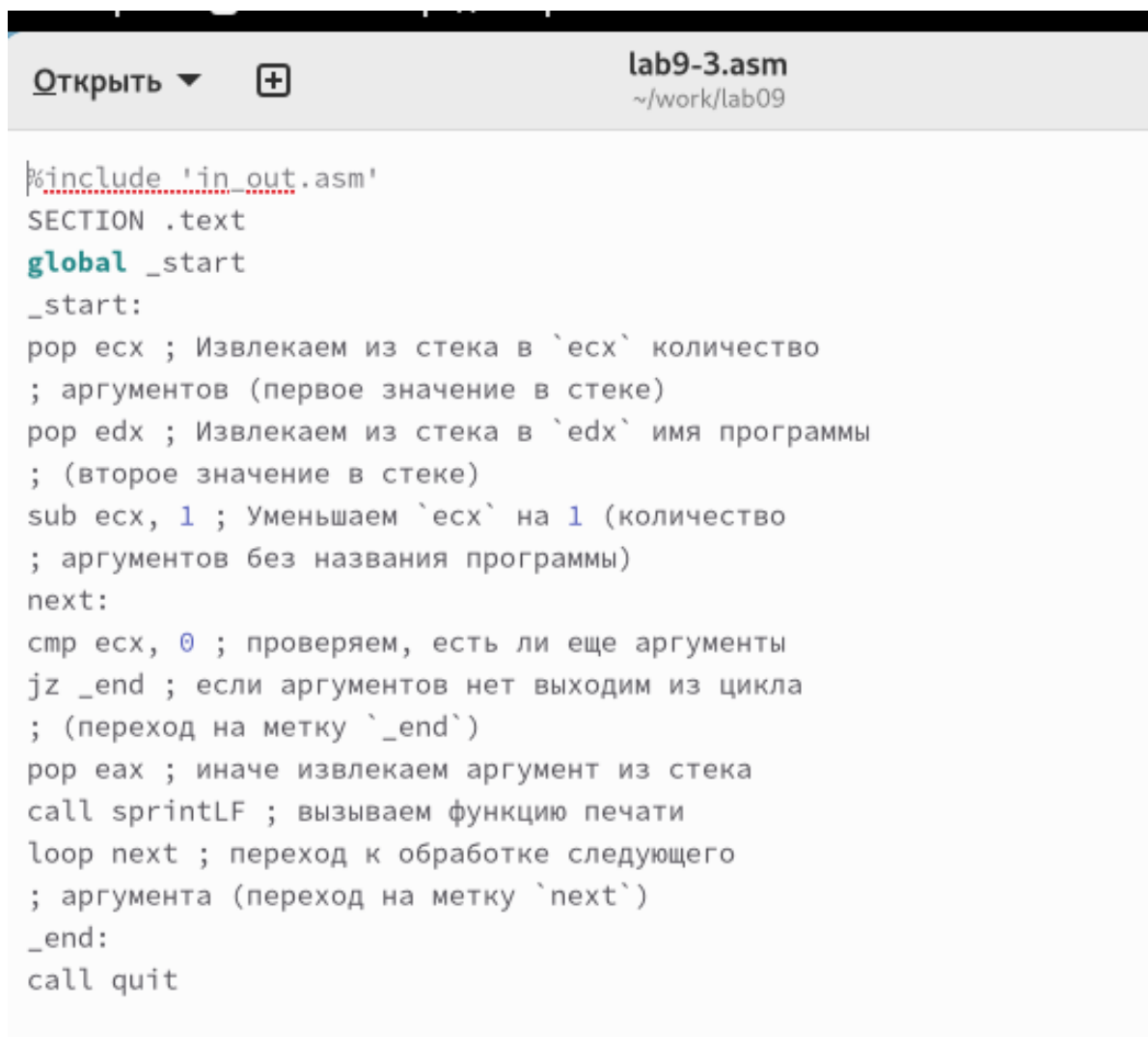
```
B+ 0x8049000 <_start>    mov    eax,0x4
    0x8049005 <_start+5>  mov    ebx,0x1
    0x804900a <_start+10> mov    ecx,0x804a000
    0x804900f <_start+15> mov    edx,0x8
    0x8049014 <_start+20> int     0x80
> 0x8049016 <_start+22> mov    eax,0x4
    0x804901b <_start+27> mov    ebx,0x1
    0x8049020 <_start+32> mov    ecx,0x804a008
    0x8049025 <_start+37> mov    edx,0x7
```

. The bottom section shows the GDB prompt and a series of commands and their outputs:

```
native process 2757 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. 2.14: Вывод значения регистра

Я скопировал файл lab8-2.asm, созданный в ходе выполнения лабораторной работы №8, который выводит аргументы командной строки, и создал исполняемый файл. Для загрузки программы с аргументами в отладчик GDB использовал ключ `-args`, указав соответствующие аргументы. Затем установил точку остановки перед первой инструкцией в программе и запустил ее.




```
Открыть ▾  lab9-3.asm  
~/work/lab09  
  
%include 'in_out.asm'  
SECTION .text  
global _start  
_start:  
pop ecx ; Извлекаем из стека в `ecx` количество  
; аргументов (первое значение в стеке)  
pop edx ; Извлекаем из стека в `edx` имя программы  
; (второе значение в стеке)  
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество  
; аргументов без названия программы)  
next:  
cmp ecx, 0 ; проверяем, есть ли еще аргументы  
jz _end ; если аргументов нет выходим из цикла  
; (переход на метку `_end`)  
pop eax ; иначе извлекаем аргумент из стека  
call sprintf ; вызываем функцию печати  
loop next ; переход к обработке следующего  
; аргумента (переход на метку `next`)  
_end:  
call quit
```

Рис. 2.15: Изменение кода

Адрес вершины стека хранится в регистре `esp`, и по этому адресу располагается число, равное количеству аргументов командной строки, включая имя программы. В данном случае число аргументов равно 5: имя программы `lab9-3` и аргументы: `аргумент1`, `аргумент2` и `‘аргумент 3’`.

Я также просмотрел остальные позиции стека. По адресу `[esp+4]` находится адрес в памяти, где хранится имя программы, по адресу `[esp+8]` хранится адрес первого аргумента, по адресу `[esp+12]` - второго и так далее.

```
malitvinov@fedora:~/work/lab09 — gdb --args lab9-3 argument 1 argument 2 argum...
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.

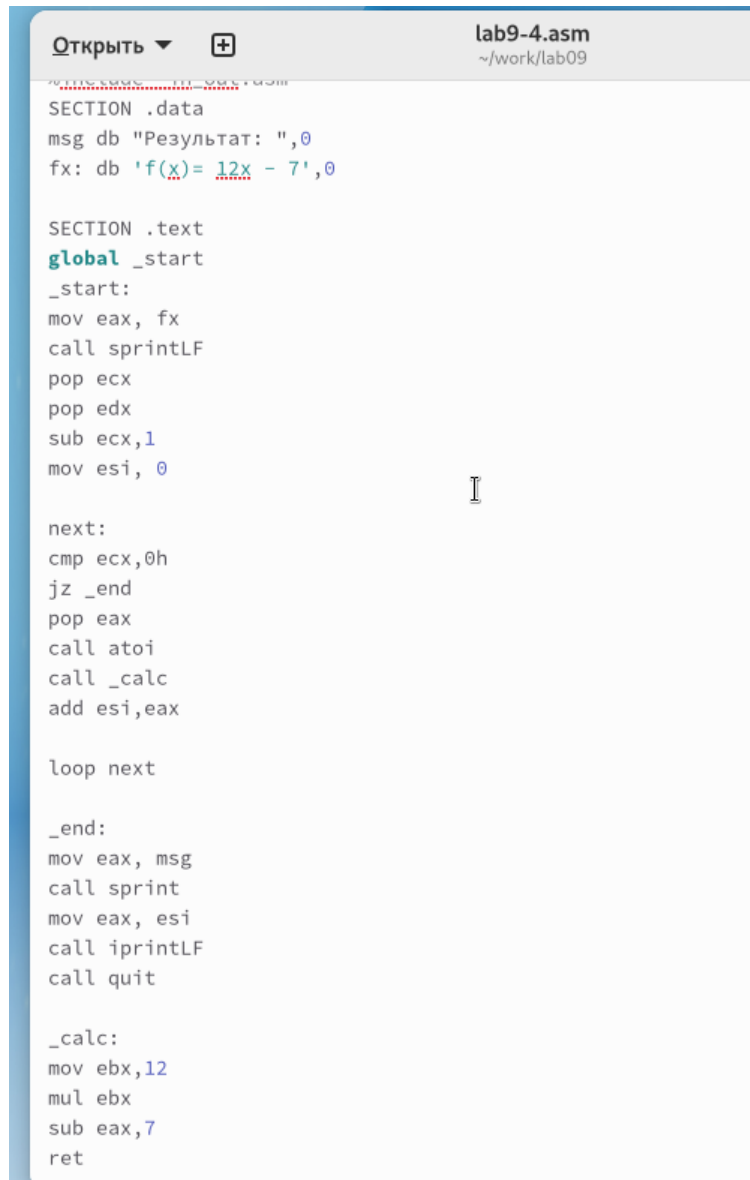
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd200:    0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd3b1:    "/home/malitvinov/work/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3d4:    "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3dd:    "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3df:    "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3e8:    "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3ea:    "argument 3"
(gdb) c
Continuing.
argument
1
argument
2
argument 3
[Inferior 1 (process 2820) exited normally]
(gdb)
```

Рис. 2.16: Вывод значения регистра

Объясню, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12]). Это связано с тем, что шаг равен размеру переменной, который составляет 4 байта.

2.3 Задание для самостоятельной работы

Я внес изменения в программу из лабораторной работы №8, чтобы вычислить значение функции $f(x)$ в виде подпрограммы.



```
Открыть ▾ + lab9-4.asm
~/work/lab09

SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 12x - 7',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

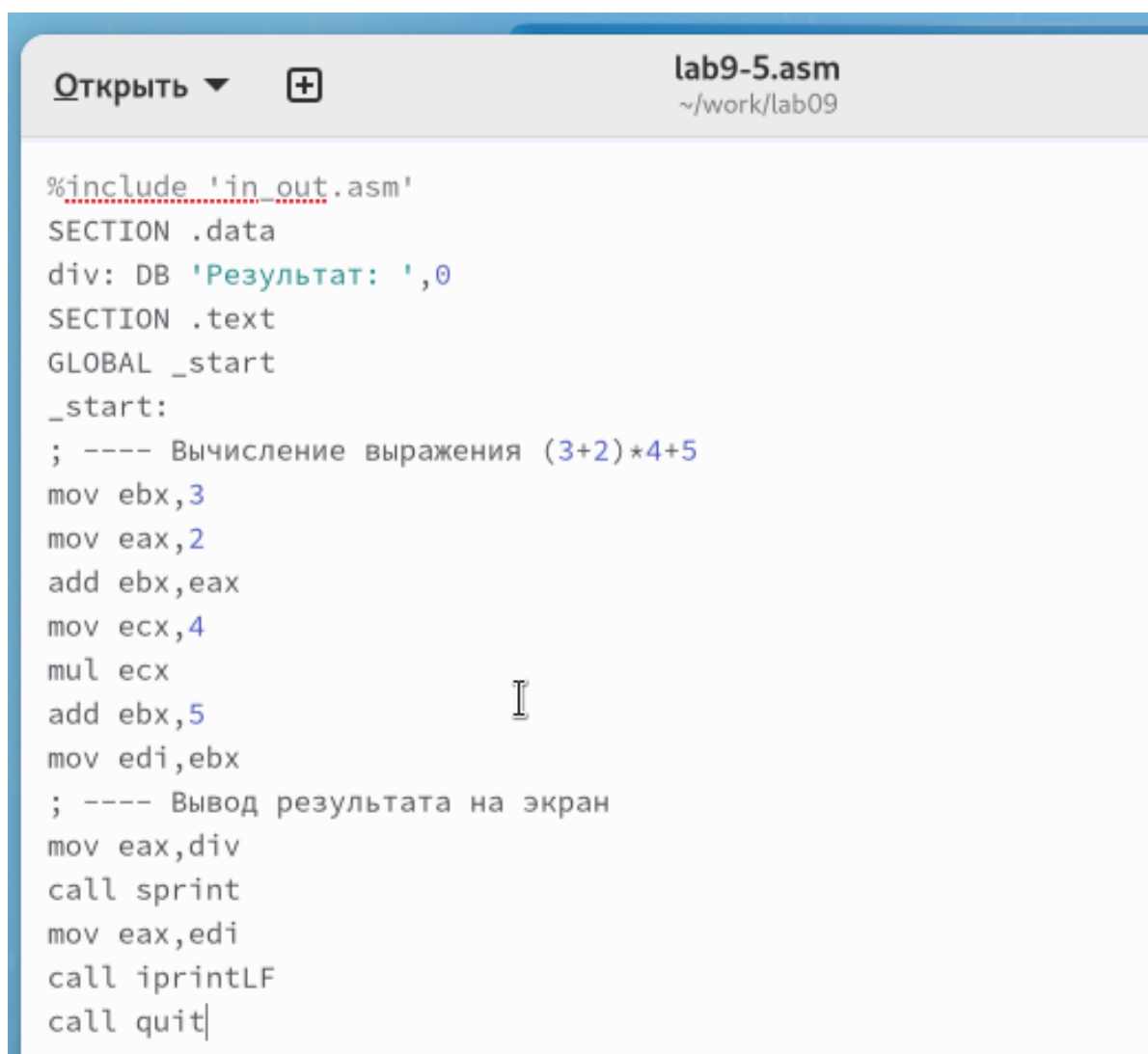
_calc:
mov ebx,12
mul ebx
sub eax,7
ret
```

Рис. 2.17: Изменение кода

```
[malitvinov@fedora lab09]$  
[malitvinov@fedora lab09]$ nasm -f elf lab9-4.asm  
[malitvinov@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o  
[malitvinov@fedora lab09]$ ./lab9-4  
f(x)= 12x - 7  
Результат: 0  
[malitvinov@fedora lab09]$ ./lab9-4 1 2 3 5  
f(x)= 12x - 7  
Результат: 104  
[malitvinov@fedora lab09]$
```

Рис. 2.18: Запуск программы

Ниже приведен исправленный листинг программы, который вычисляет выражение $(3 + 2) * 4 + 5$. Однако, при запуске, программа дает неверный результат. Я решил использовать отладчик GDB для анализа изменений значений регистров и определения ошибки.




```
Открыть ▾  lab9-5.asm  
~/work/lab09  
  
%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. 2.19: Код с ошибкой

The screenshot shows a GDB window titled "malitvinov@fedora:~/work/lab09 — gdb lab9-5". The top panel displays register values:

Register	Value	Comment
eax	0x8	8
ecx	0x4	4
edx	0x0	0
ebx	0xa	10
esp	0xffffd230	0xffffd230
ebp	0x0	0x0
esi	0x0	0
edi	0xa	10

The middle panel shows assembly code with a blue selection box around the first six lines:

```
B+ 0x80490e8 <_start>    mov     ebx,0x3
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>
```

Below the assembly code, a memory dump shows the value "04a000" at address "04a000".

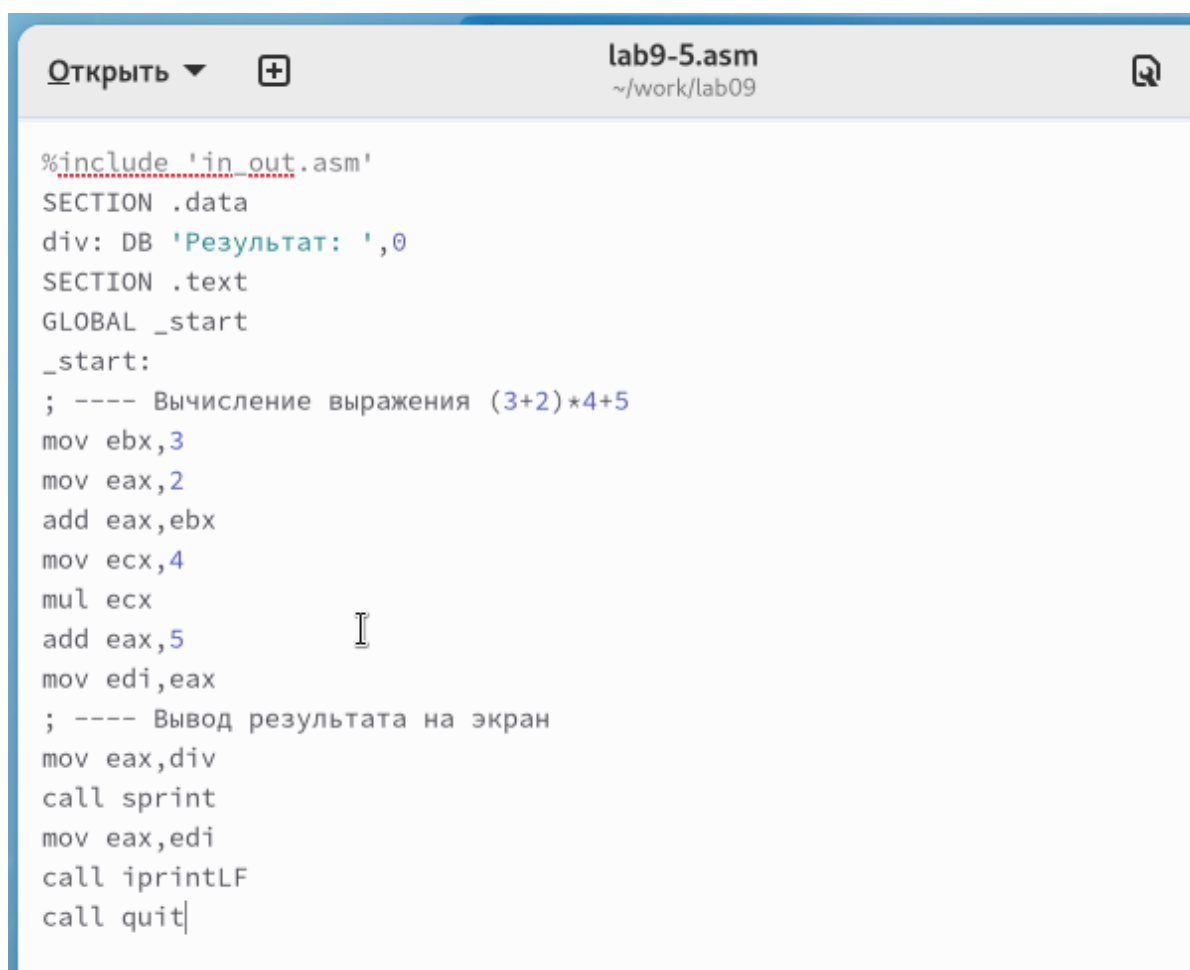
The bottom panel shows the GDB command line and status:

```
native process 3883 In: _start L16 PC: 0x8049100
Breakpoint 1: No process In: 9-5.asm:8 L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 3883) exited normally]
(gdb)
```

Рис. 2.20: Отладка

В процессе отладки я заметил, что порядок аргументов в инструкции add был перепутан, и что при завершении работы, вместо eax, значение отправлялось в edi.

Вот исправленный код программы:



```
Открыть ▾ + lab9-5.asm  
~/work/lab09  
  
%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. 2.21: Код исправлен

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

eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd230 0xffffd230
ebp      0x0       0x0
esi      0x0       0
edi      0x0       0

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490fe <_start+22>    mov     edi,eax
0x8049100 <_start+24>    add     eax,eb804a000
0x8049105 <_start+29>    call    0x804900f <sprint>
0x804910a <_start+34>    mul     eax,edi
0x804910c <_start+36>    call    0x8049086 <iprintLF>
> 0x8049111 <_start+41>    call    0x80490db <quit>
                                04a000
                                rint>

native process 3926 In: _start L14 PC: 0x80490fe
No process In: L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 3926) exited normally]
(gdb)
```

Рис. 2.22: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.