

# **Отчёт по лабораторной работе №9**

**Дисциплина: Архитектура компьютера**

**Бережной Иван Александрович**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Реализация программ в NASM . . . . .	7
3.2	Отладка программ с помощью GDB . . . . .	11
3.3	Задание для самостоятельной работы . . . . .	23
<b>4</b>	<b>Выводы</b>	<b>32</b>
	<b>Список литературы</b>	<b>33</b>

# Список иллюстраций

3.1	Создание рабочего каталога . . . . .	7
3.2	Копирование листинга в lab09-1.asm . . . . .	8
3.3	Проверка работы исполняемого файла lab09-1 . . . . .	9
3.4	Изменение программы . . . . .	10
3.5	Проверка работы изменённой программы . . . . .	11
3.6	Копирование листинга в lab09-2.asm . . . . .	11
3.7	Загрузка исполняемого файла в отладчик . . . . .	12
3.8	Установка брейкпоинта _start . . . . .	12
3.9	Сравнение отображений команд . . . . .	14
3.10	Включение режима псевдографики . . . . .	15
3.11	Просмотр списка точек останова . . . . .	15
3.12	Установка брейкпоинта по адресу инструкции . . . . .	16
3.13	Начальные значения регистров . . . . .	17
3.14	Новые значения регистров . . . . .	18
3.15	Просмотр содержимого переменных . . . . .	19
3.16	Замена символов в переменных . . . . .	19
3.17	Вывод значения регистра . . . . .	20
3.18	Изменение значения регистра . . . . .	21
3.19	Создание файла lab09-3.asm . . . . .	21
3.20	Загрузка исполняемого файла в отладчик с аргументами . . . . .	22
3.21	Установка брейкпоинта . . . . .	22
3.22	Просмотр позиций стека . . . . .	23
3.23	Написание программы . . . . .	24
3.24	Тестирование программы . . . . .	24
3.25	Копирование листинга в lab9-5.asm . . . . .	26
3.26	Создание исполняемого файла lab09-5.exe . . . . .	27
3.27	Начало проверки программы . . . . .	28
3.28	Конец проверки программы . . . . .	29
3.29	Исправление кода . . . . .	30
3.30	Проверка исправленного кода . . . . .	30

## **Список таблиц**

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Задание

1. Реализация программ в NASM
2. Отладка программ с помощью GDB
3. Задание для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация программ в NASM

Создадим каталог для дальнейшего выполнения лабораторной работы. Перейдём в него и создадим файл lab09-1.asm (рис. 3.1).

```
[iaberezhnoy@fedora ~]$ mkdir ~/work/arch-pc/lab09  
[iaberezhnoy@fedora ~]$ cd ~/work/arch-pc/lab09  
[iaberezhnoy@fedora lab09]$ touch lab09-1.asm
```

Рис. 3.1: Создание рабочего каталога

Скопируем код из предложенного листинга 9.1. (рис. 3.2), создадим исполняемый файл и проверим его работу (рис. 3.3).

```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab09
lab09-1.asm [----] 0 L: [ 1+35 36/ 36] *(748 / 748b) <EOF>
#include 'in_out.asm'
SECTION .data
    msg: DB 'Введите x: ',0
    result: DB '2x+7=',0
SECTION .bss
    x: RESB 80
    res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
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, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret ; выход из подпрограммы
```

Рис. 3.2: Копирование листинга в lab09-1.asm



```
[iaberezhnoy@fedora lab09]$ nasm -f elf lab09-1.asm
[iaberezhnoy@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[iaberezhnoy@fedora lab09]$ ./lab09-1
Введите x: 5
2x+7=17
[iaberezhnoy@fedora lab09]$ ./lab09-1
Введите x: 43
2x+7=93
[iaberezhnoy@fedora lab09]$
```

Рис. 3.3: Проверка работы исполняемого файла lab09-1

Изменим текст программы, добавив подпрограмму, которая вычисляет значение выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры (рис. 3.4). Создадим исполняемый файл и проверим его работу (рис. 3.5).

```

mc [iaberezhnoy@fedora]:~/work/arch-pc/lab09
lab09-1.asm [-M--] 9 L:[ 1+19 20/ 42] *(390 / 837b) 0010 0x00A
%include 'in_out.asm'
SECTION .data
    msg: DB 'Введите x: ',0
    result: DB '2(3x-1)+7=',0
SECTION .bss
    x: RESB 80
    res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
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, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
    call _subcalcul
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret ; выход из подпрограммы
....
_subcalcul:
    mov ebx, 3
    mul ebx
    sub eax, 1
    ret

```

Рис. 3.4: Изменение программы

```

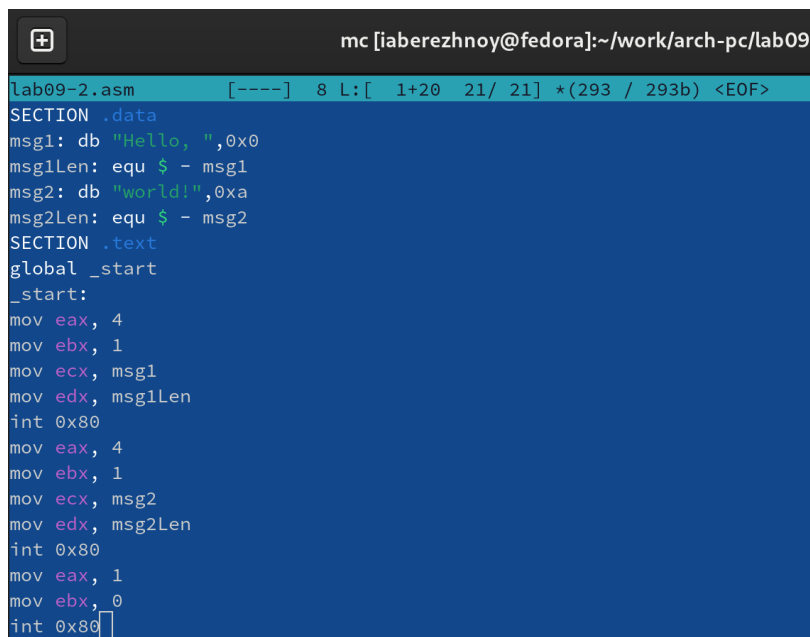
[iaberezhnoy@fedora lab09]$ nasm -f elf lab09-1.asm
[iaberezhnoy@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[iaberezhnoy@fedora lab09]$ ./lab09-1
Введите x: 1
2(3x-1)+7=11
[iaberezhnoy@fedora lab09]$ ./lab09-1
Введите x: 2
2(3x-1)+7=17

```

Рис. 3.5: Проверка работы изменённой программы

## 3.2 Отладка программ с помощью GDB

Создадим файл lab09-2.asm и скопируем в него листинг 9.2. (рис. 3.6), затем создадим исполняемый файл, загрузим его в отладчик GDB и проверим его работу (рис. 3.7).



```

mc [iaberezhnoy@fedora]:~/work/arch-pc/lab09
lab09-2.asm  [----]  8 L: [ 1+20 21/ 21] *(293 / 293b) <EOF>
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. 3.6: Копирование листинга в lab09-2.asm

```

[iaberezhnoy@fedora lab09]$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
[iaberezhnoy@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o
[iaberezhnoy@fedora lab09]$ gdb lab09-2
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 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 lab09-2...
(gdb) run
Starting program: /home/iaberezhnoy/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Hello, world!
[Inferior 1 (process 4204) exited normally]

```

Рис. 3.7: Загрузка исполняемого файла в отладчик

Установим точку останова командой `break _start` и запустим программу (рис. 3.8).

```

(gdb) break _start
Breakpoint 1 at 0x4010e0: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/iaberezhnoy/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4

```

Рис. 3.8: Установка брейкпоинта `_start`

Посмотрим на дисассимилированный код программы командой `disassemble`, начиная с брейкпоинта. Теперь переключим отобра-

жение команд на Intel'овский синтаксис с помощью команды `set disassembly-flavour intel`, посмотрим и на это отображение (рис. 3.9). Можем заметить различия: 1. В отображении АТТ имена регистров начинаются с символов `%` или `$`, в отличии от отображения Intel, где перед регистрами символов нет; 2. В отображении АТТ другой порядок регистров, нежели в отображении Intel.

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x004010e0 <+0>:      mov     $0x4,%eax
    0x004010e5 <+5>:      mov     $0x1,%ebx
    0x004010ea <+10>:     mov     $0x402118,%ecx
    0x004010ef <+15>:     mov     $0x8,%edx
    0x004010f4 <+20>:     int     $0x80
    0x004010f6 <+22>:     mov     $0x4,%eax
    0x004010fb <+27>:     mov     $0x1,%ebx
    0x00401100 <+32>:     mov     $0x402120,%ecx
    0x00401105 <+37>:     mov     $0x7,%edx
    0x0040110a <+42>:     int     $0x80
    0x0040110c <+44>:     mov     $0x1,%eax
    0x00401111 <+49>:     mov     $0x0,%ebx
    0x00401116 <+54>:     int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x004010e0 <+0>:      mov     eax,0x4
    0x004010e5 <+5>:      mov     ebx,0x1
    0x004010ea <+10>:     mov     ecx,0x402118
    0x004010ef <+15>:     mov     edx,0x8
    0x004010f4 <+20>:     int     0x80
    0x004010f6 <+22>:     mov     eax,0x4
    0x004010fb <+27>:     mov     ebx,0x1
    0x00401100 <+32>:     mov     ecx,0x402120
    0x00401105 <+37>:     mov     edx,0x7
    0x0040110a <+42>:     int     0x80
    0x0040110c <+44>:     mov     eax,0x1
    0x00401111 <+49>:     mov     ebx,0x0
    0x00401116 <+54>:     int     0x80
End of assembler dump.

```

Рис. 3.9: Сравнение отображений команд

Включим режим псевдографики для удобного анализа программы командами `layout asm` и `layout regs` (рис. 3.10).

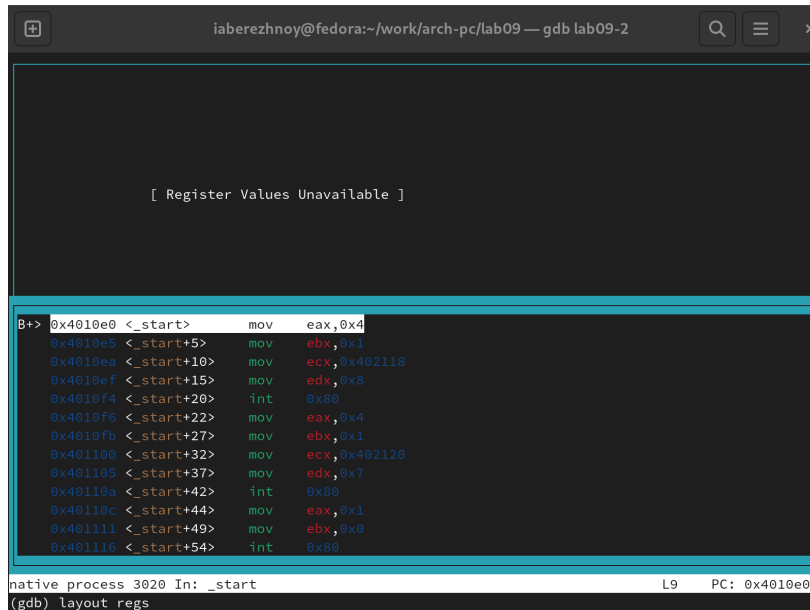


Рис. 3.10: Включение режима псевдографики

Посмотрим, числится ли уже установленный нами брейкпоинт в списке точек останова командой `info breakpoints` (рис. 3.11). Установим вторую точку останова по адресу инструкции. Для этого пишем команду `break *` и после символа звёздочки без пробела вводим адрес нужной нам инструкции (а именно `mov ebx,0x0`). Снова посмотрим список точек останова (рис. 3.12).

```

(gdb) info breakpoints
Num      Type           Disp Enb Address          What
1        breakpoint    keep y   0x004010e0 lab09-2.asm:9
        breakpoint already hit 1 time

```

Рис. 3.11: Просмотр списка точек останова

```

(gdb) break *0x401111
Breakpoint 2 at 0x401111: file lab09-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x004010e0 lab09-2.asm:9
2        breakpoint     keep y   0x00401111 lab09-2.asm:20

```

Рис. 3.12: Установка брейкпоинта по адресу инструкции

Выполним команду `si 5`, которая позволит пошагово выполнять инструкции. Теперь посмотрим, какие регистры были изменены, сравнив их начальные значения (рис. 3.13) с новыми (рис. 3.14). Были изменены значения в регистрах `eax`, `ebx`, `ecx`, `edx`, `eip`, `cs`, `ds`, `ss`, `eflags` и `es`.



```
iaberezhnuy@fedora:~/work/arch-pc/lab09 — gdb lab09-2

--Register group: general--
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x4010e0 0x4010e0 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

B+> 0x4010e0 <_start> mov eax,0x4
0x4010e5 <_start+5> mov ebx,0x1
0x4010ea <_start+10> mov ecx,0x402118
0x4010ef <_start+15> mov edx,0x8
0x4010f4 <_start+20> int 0x80
0x4010f6 <_start+22> mov eax,0x4
0x4010fb <_start+27> mov ebx,0x1
0x401100 <_start+32> mov ecx,0x402120
0x401105 <_start+37> mov edx,0x7
0x40110a <_start+42> int 0x80
0x40110c <_start+44> mov eax,0x1
b+ 0x401111 <_start+49> mov ebx,0x0
0x401116 <_start+54> int 0x80

native process 3146 In: _start L9 PC: 0x4010e0
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x4010e0 0x4010e0 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 3.13: Начальные значения регистров

```
iaberezhnoy@fedora:~/work/arch-pc/lab09 — gdb lab09-2
--Register group: general--
eax      0x8      8
ecx      0x402118 4202776
edx      0x8      8
ebx      0x1      1
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x4010f6 0x4010f6 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

B+ 0x4010e0 <_start>    mov     eax,0x4
0x4010e5 <_start+5>    mov     ebx,0x1
0x4010ea <_start+10>   mov     ecx,0x402118
0x4010ef <_start+15>   mov     edx,0x8
0x4010f4 <_start+20>   int     0x80
> 0x4010f6 <_start+22> mov     eax,0x4
0x4010fb <_start+27>   mov     ebx,0x1
0x401100 <_start+32>   mov     ecx,0x402120
0x401105 <_start+37>   mov     edx,0x7
0x40110a <_start+42>   int     0x80
0x40110c <_start+44>   mov     eax,0x1
b+ 0x401111 <_start+49> mov     ebx,0x0
0x401116 <_start+54>   int     0x80

native process 3146 In: _start L14 PC: 0x4010f6
edx      0x0      0
ebx      0x0      0
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x4010e0 0x4010e0 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--qQuit
(gdb) si 5
(gdb) 
```

Рис. 3.14: Новые значения регистров

Посмотрим значение переменной `msg1` по имени с помощью команды `x/1sb &msg1`. Сделаем то же самое для переменной `msg2`, но уже по адресу (рис. 3.15). Изменим первый символ переменной `msg1` командой `'set {char}msg1='h'` и посмотрим, сработали изменения. Также изменим первый символ и во второй переменной (рис. 3.16).

```

(gdb) x/1sb &msg1
0x402118 <msg1>:      "Hello, "
(gdb) x/1sb 0x402120
0x402120 <msg2>:      "world!\n\034"

```

Рис. 3.15: Просмотр содержимого переменных

```

(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x402118 <msg1>:      "hello, "
(gdb) set {char}&msg2='W'
(gdb) x/1sb &msg2
0x402120 <msg2>:      "World!\n\034"

```

Рис. 3.16: Замена символов в переменных

Выведем значение регистра `edx` в различных форматах командами `p/F $<регистр>` (рис. 3.17).

```
(gdb) p/x $edx
$1 = 0x8

(gdb) p/t $edx
$2 = 1000

(gdb) p/c $edx
$3 = 8 '\b'
```

Рис. 3.17: Вывод значения регистра

Командой `set $ebx=<value>` изменим значение регистра `ebx` на `'2'`, а затем на `2`. Проверим значение после каждого его изменения (рис. 3.18). Значения различаются, хотя мы ввели одинаковые числа. Тем не менее, в первом случае мы вводили число как символ, соответственно в качестве значения видим код этого числа (50). Во втором же случае мы вводим число как числовой тип данных, поэтому и получили то же самое число.

```
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$4 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$5 = 2
```

Рис. 3.18: Изменение значения регистра

Завершим выполнение программы и выйдем из отладчика GDB.

Создадим файл lab09-3.asm и скопируем в него код из файла lab8-2.asm. Создадим исполняемый файл (рис. 3.19). Теперь загрузим исполняемый файл в отладчик, попутно указав нужные аргументы (для этого также нужно использовать ключ `--args` (рис. 3.20)).

```
[iaberezhnuy@fedora lab09]$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm  
[iaberezhnuy@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm  
[iaberezhnuy@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
```

Рис. 3.19: Создание файла lab09-3.asm

```
[iaberezhnoy@fedora lab09]$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 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 lab09-3...
(gdb) █
```

Рис. 3.20: Загрузка исполняемого файла в отладчик с аргументами

Установим брейкпоинт и запустим программу (рис. 3.21). Посмотрим позиции стека через регистр `esp` командой `x/s *(void**)($esp + <value>)` (рис. 3.22). Шаг изменения равен 4, т.к. шаг - `int`, под который выделяется 4 байта памяти.

```
(gdb) b _start
Breakpoint 1 at 0x4011a8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/iaberezhnoy/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) █
```

Рис. 3.21: Установка брейкпоинта

```

(gdb) x/x $esp
0xffffd170:    0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd311:    "/home/iaberezhnoy/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd33e:    "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd350:    "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd361:    "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd363:    "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:    <error: Cannot access memory at address 0x0>

```

Рис. 3.22: Просмотр позиций стека

### 3.3 Задание для самостоятельной работы

1. Скопируем код программы из задания для самостоятельной работы (лабораторная работы №8) в файл lab09-4.asm. Перепишем код так, чтобы вычисление функции происходило в подпрограмме (функция соответствует варианту 2 из лабораторной работы №8) (рис. 3.23). Создадим исполняемый файл и проверим его работу (рис. 3.24).

```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab09
lab09-4.asm [----] 0 L:[ 1+ 0 1/ 38] *(0 / 427b) 0037 0x025
%include 'in_out.asm'

SECTION .data
    msg db "Результат: ", 0

SECTION .text
global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0
    ....

next:
    cmp ecx, 0h
    jz_end
    pop eax
    call atoi
    call solve
    loop next
    ....

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

solve:
    imul eax, 3
    sub eax, 1
    add esi, eax
    ret
```

Рис. 3.23: Написание программы

```
[iaberezhnoy@fedora lab09]$ nasm -f elf lab09-4.asm
lab09-4.asm:18: warning: label alone on a line without a colon m
[iaberezhnoy@fedora lab09]$ ld -m elf_i386 -o lab09-4 lab09-4.o
[iaberezhnoy@fedora lab09]$ ./lab09-4 1 2 3
Результат: 15
[iaberezhnoy@fedora lab09]$ ./lab09-4 1
Результат: 2
```

Рис. 3.24: Тестирование программы



### Листинг 9.1. Программа нахождения значения функции с использованием подпрограммы

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg db "Результат: ", 0
```

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
    pop ecx
```

```
    pop edx
```

```
    sub ecx, 1
```

```
    mov esi, 0
```

```
next:
```

```
    cmp ecx, 0h
```

```
    jz_end
```

```
    pop eax
```

```
    call atoi
```

```
    call solve
```

```
    loop next
```

```
_end:
```

```

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

```

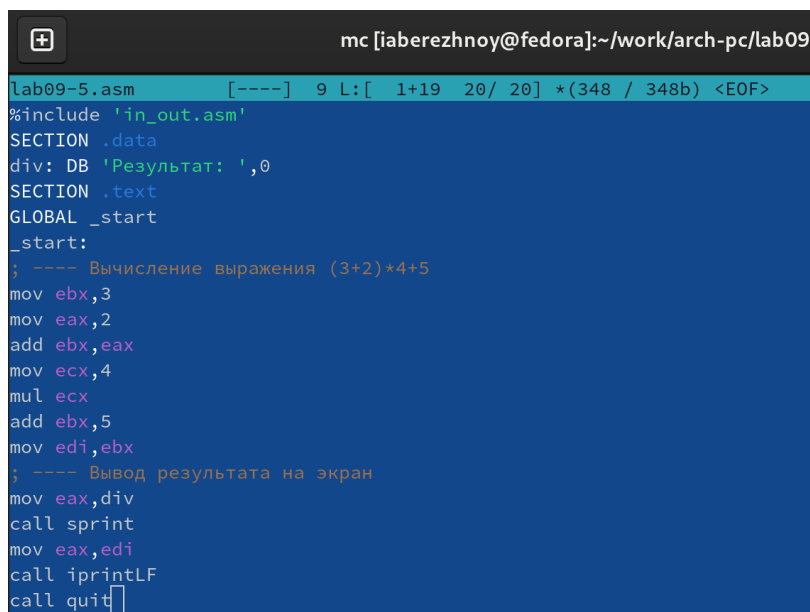
solve:

```

imul eax, 3
sub eax, 1
add esi, eax
ret

```

2. Создадим файл lab09-5.asm и скопируем в него предложенный листинг 9.3 (рис. 3.25).



```

mc [iaberezhnoy@fedora]:~/work/arch-pc/lab09
lab09-5.asm  [----]  9 L:[ 1+19 20/ 20] *(348 / 348b) <EOF>
#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. 3.25: Копирование листинга в lab9-5.asm

Создадим исполняемый файл, загрузим его в отладчик GDB и проверим его работу (рис. 3.26). Результат получился неверным. Установим несколько брейкпоинтов и пройдемся по коду, учитывая изменения значений регистров (рис. 3.27 и рис. 3.28).

```
[iaberezhnuy@fedora lab09]$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
[iaberezhnuy@fedora lab09]$ ld -m elf_i386 -o lab09-5 lab09-5.o
[iaberezhnuy@fedora lab09]$ gdb lab09-5
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 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 lab09-5...
```

Рис. 3.26: Создание исполняемого файла lab09-5.exe

```
iaberezhnoy@fedora:~/work/arch-pc/lab09 — gdb lab09-5
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x3      3
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x4011cd 0x4011cd <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x4011c9 <_start>      mov     ebx,0x3
B> 0x4011cd <_start+5>  mov     eax,0x2
b+ 0x4011d2 <_start+10> add     ebx,eax
0x4011d4 <_start+12>  mov     ecx,0x4
b+ 0x4011d9 <_start+17> mul     ecx
b+ 0x4011db <_start+19> add     ebx,0x5
0x4011de <_start+22>  mov     edi,ebx
0x4011e0 <_start+24>  mov     eax,0x4021f8
0x4011e5 <_start+29>  call    0x4010ef <sprint>
0x4011ea <_start+34>  mov     eax,edi
0x4011ec <_start+36>  call    0x401166 <iprintLF>
0x4011f1 <_start+41>  call    0x4011bb <quit>
0x4011f6          add     BYTE PTR [eax],al

native No process in:                                L??  PC: ??
native process 3916 In: _start                        L9   PC: 0x4011cd
(gdb) break 0x4011cd
Function "0x4011cd" not defined.
(gdb) break *0x4011cd
Breakpoint 1 at 0x4011cd: file lab09-5.asm, line 9.
(gdb) break *0x4011d2
Breakpoint 2 at 0x4011d2: file lab09-5.asm, line 10.
(gdb) break *0x4011d9
Breakpoint 3 at 0x4011d9: file lab09-5.asm, line 12.
(gdb) break *0x4011db
Breakpoint 4 at 0x4011db: file lab09-5.asm, line 13.
(gdb) run
Starting program: /home/iaberezhnoy/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:9
(gdb) 
```

Рис. 3.27: Начало проверки программы

```
iaberezhnoy@fedora:~/work/arch-pc/lab09 — gdb lab09-5
Register group: general
eax      0x2      2
ecx      0x0      0
edx      0x0      0
ebx      0x3      3
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x4011d2 0x4011d2 <_start+10>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x4011c8 <_start>      mov     ebx,0x3
B+ 0x4011cd <_start+5>  mov     eax,0x2
B+> 0x4011d2 <_start+10> add     ebx,eax
0x4011d4 <_start+12>  mov     ecx,0x4
b+ 0x4011d9 <_start+17> mul     ecx
b+ 0x4011db <_start+19> add     ebx,0x5
0x4011de <_start+22>  mov     edi,ebx
0x4011e0 <_start+24>  mov     eax,0x4021f8
0x4011e5 <_start+29>  call    0x4018ef <sprint>
0x4011ea <_start+34>  mov     eax,edi
0x4011ec <_start+36>  call    0x401166 <iprintf>
0x4011f1 <_start+41>  call    0x4011bb <quit>
0x4011f6             add     BYTE PTR [eax],al

native No process in: L?? PC: ??
native process 3916 In: _start L10 PC: 0x4011d2
Breakpoint 1 at 0x4011cd: file lab09-5.asm, line 9.
(gdb) break *0x4011d2
Breakpoint 2 at 0x4011d2: file lab09-5.asm, line 10.
(gdb) break *0x4011d9
Breakpoint 3 at 0x4011d9: file lab09-5.asm, line 12.
(gdb) break *0x4011db
Breakpoint 4 at 0x4011db: file lab09-5.asm, line 13.
(gdb) run
Starting program: /home/iaberezhnoy/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:9
(gdb) si

Breakpoint 2, _start () at lab09-5.asm:10
(gdb) 
```

Рис. 3.28: Конец проверки программы

Очевидно, что присутствуют несколько ошибок, которые мы исправим (рис. 3.29). Проверим работу программы (рис. 3.30).

```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab09
lab09-5.asm [----] 10 L: [ 1+ 9 10/ 21] *(192 / 349b) 0120 0x07
#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. 3.29: Исправление кода

```
[iaberezhnoy@fedora lab09]$ nasm -f elf lab09-5.asm
[iaberezhnoy@fedora lab09]$ ld -m elf_i386 -o lab09-5 lab09-5.o
[iaberezhnoy@fedora lab09]$ ./lab09-5
Результат: 25
```

Рис. 3.30: Проверка исправленного кода

## Листинг 9.2. Исправленная программа, находящая значение выражения

```
%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

```

## 4 Выводы

В ходе выполнения лабораторной работы мы приобрели навыки написания программ с использованием подпрограмм и познакомились с методами отладки при помощи GDB и его основными возможностями.



# Список литературы

::: Архитектура ЭВМ