

Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Луцкая Алиса Витальевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Релаксация подпрограмм в NASM	8
4.1.1	Отладка программ с помощью GDB	11
4.1.2	Добавление точек останова	16
4.1.3	Работа с данными программы в GDB	17
4.1.4	Обработка аргументов командной строки в GDB	22
4.2	Задание для самостоятельной работы	23
5	Выводы	27
6	Список литературы	28

Список иллюстраций

4.1	Создание файла	8
4.2	Ввод программы	9
4.3	Запуск программы из листинга	9
4.4	Изменение программы первого листинга	10
4.5	Запуск программы из листинга	11
4.6	Создание файла	11
4.7	Загрузка файла	12
4.8	Запуск программы	12
4.9	Запуск отладчика с брейкпоинтом	13
4.10	Просмотр кода	13
4.11	Переключение	14
4.12	Дисассимилирование программы	15
4.13	Режим псевдографики	16
4.14	Добавление второй точки останова	17
4.15	Просмотр содержимого регистров	18
4.16	Просмотр содержимого переменных двумя способами	19
4.17	Изменение содержимого переменных двумя способами	20
4.18	Просмотр значения регистра разными представлениями	21
4.19	Примеры использования команды set	22
4.20	Подготовка новой программы	22
4.21	Проверка работы стека	23
4.22	Измененная программа предыдущей лабораторной работы	24
4.23	Поиск ошибки в программе через пошаговую отладку	25
4.24	Исправление ошибки	26
4.25	Проверка корректировок в программе	26

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 подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Задания для самостоятельной работы

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

- обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки.

Можно выделить следующие типы ошибок:

- синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отработывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

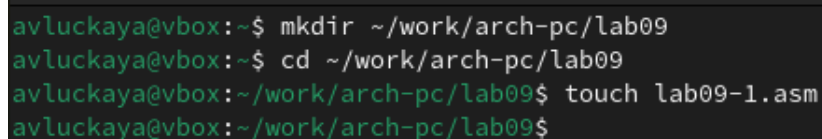
Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

4 Выполнение лабораторной работы

4.1 Релазация подпрограмм в NASM

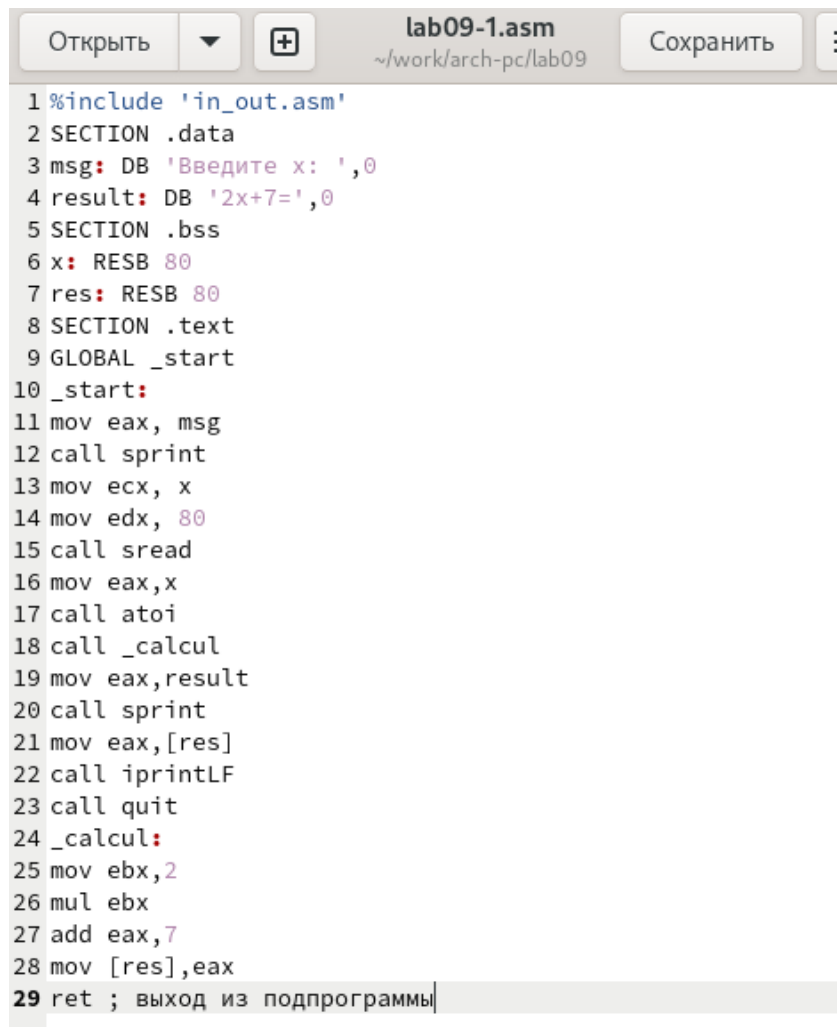
Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и создаю файл lab09-1.asm (рис. -fig. 4.1).

A screenshot of a terminal window showing four lines of commands and their outputs. The prompt is 'avluckaya@vbox:~\$'. The first command is 'mkdir ~/work/arch-pc/lab09', the second is 'cd ~/work/arch-pc/lab09', the third is 'touch lab09-1.asm', and the fourth is an empty prompt line. The output for the first three commands is '~\$' followed by the command path, and for the fourth is '~\$' followed by the file path.

```
avluckaya@vbox:~$ mkdir ~/work/arch-pc/lab09
avluckaya@vbox:~$ cd ~/work/arch-pc/lab09
avluckaya@vbox:~/work/arch-pc/lab09$ touch lab09-1.asm
avluckaya@vbox:~/work/arch-pc/lab09$
```

Рис. 4.1: Создание файла

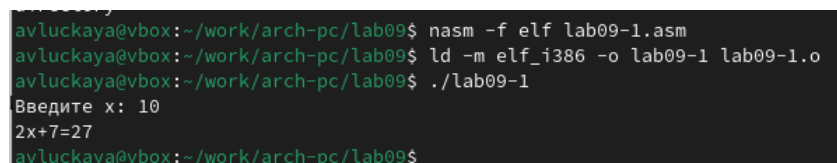
Ввод в файл кода из листинга (рис. -fig. 4.2).



```
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 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax, x
17 call atoi
18 call _calcul
19 mov eax, result
20 call sprint
21 mov eax, [res]
22 call iprintLF
23 call quit
24 _calcul:
25 mov ebx, 2
26 mul ebx
27 add eax, 7
28 mov [res], eax
29 ret ; выход из подпрограммы
```

Рис. 4.2: Ввод программы

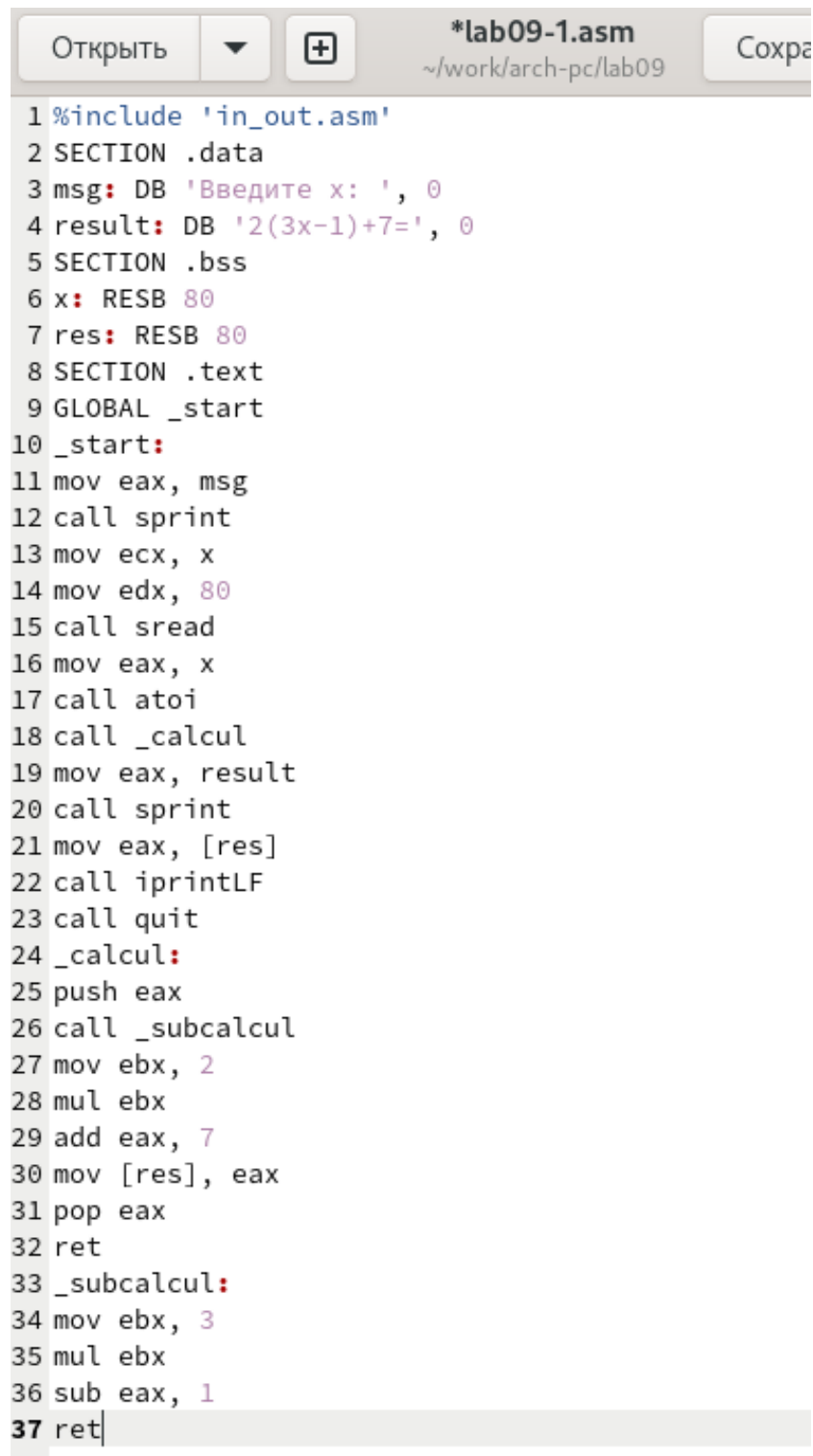
Компилирую и запускаю его, данная программа выполняет вычисление функции(рис. -fig. 4.3).



```
avluckaya@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
avluckaya@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
avluckaya@vbox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 10
2x+7=27
avluckaya@vbox:~/work/arch-pc/lab09$
```

Рис. 4.3: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. -fig. 4.4).



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

Рис. 4.4: Изменение программы первого листинга

Компилирую и запускаю его (рис. -fig. 4.5).

```

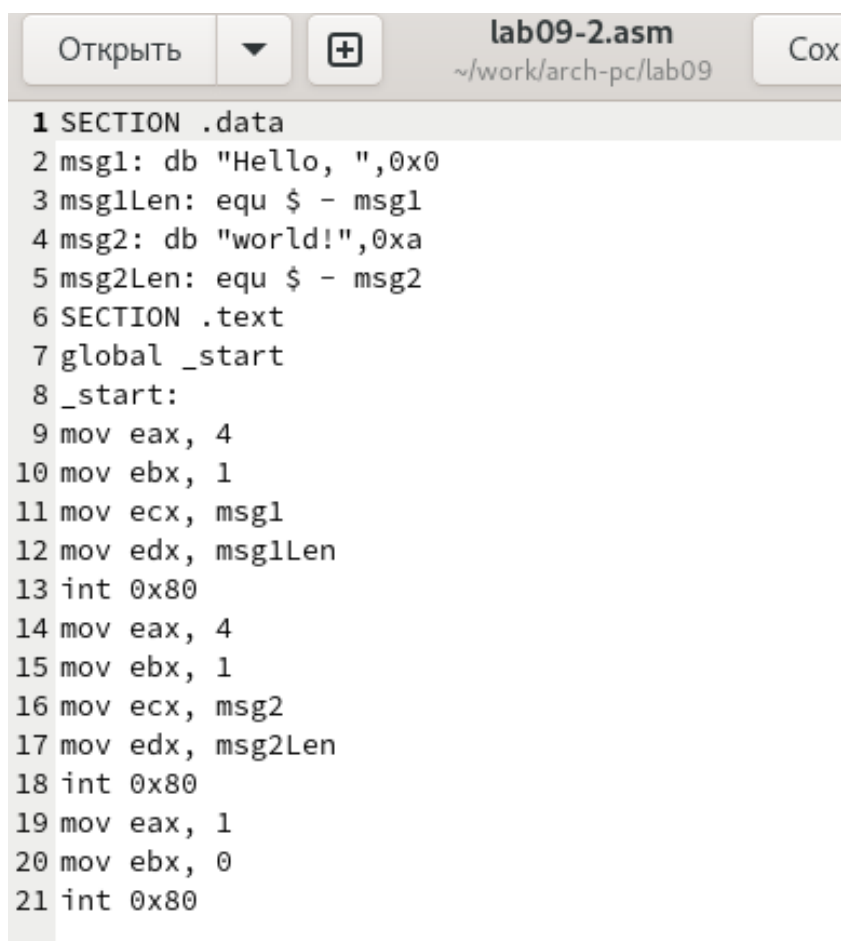
avluckaya@vbox: ~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
avluckaya@vbox: ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
avluckaya@vbox: ~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2(3x-1)+7=35

```

Рис. 4.5: Запуск программы из листинга

4.1.1 Отладка программ с помощью GDB

Создайте файл lab09-2.asm с текстом программы из Листинга 9.2.(рис. -fig. 4.6).



```

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

```

Рис. 4.6: Создание файла

Загружаю исполняемый файл в отладчик gdb (рис. -fig. 4.7).

```

avluckaya@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
avluckaya@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
avluckaya@vbox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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)

```

Рис. 4.7: Загрузка файла

Проверьте работу программы, запустив ее в оболочке GDB с помощью команды `run` (рис. -fig. 4.8).

```

avluckaya@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
avluckaya@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
avluckaya@vbox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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/avluckaya/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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 129640) exited normally]
(gdb)

```

Рис. 4.8: Запуск программы

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start` и запускаю её. (рис. -fig. 4.9).

```

avluckaya@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
avluckaya@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
avluckaya@vbox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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/avluckaya/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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 129640) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/avluckaya/work/arch-pc/lab09/lab09-2

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

```

Рис. 4.9: Запуск отладчика с брейкпоинтом

Просматриваю дисассимилированный код программы с помощью команды `disassemble` (рис. -fig. 4.10).

```

To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
Enable debuginfod for this session? (y or [n]) [Inferior 1 (process 123468) exited normally]
(gdb) run
Starting program: /home/avluckaya/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 123573) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/avluckaya/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      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. 4.10: Просмотр кода

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. -fig. 4.11).

```
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. 4.11: Переключение

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel (рис. -fig. 4.12).

Различия между синтаксисом АТТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как `ax`, `eax`, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```

(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) 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. 4.12: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. 4.13).

```

avluckaya@vbox:~/work/arch-pc/lab09—gdb lab09-2
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd0c0  0xffffd0c0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000  <_start>  eflags     0x202     [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       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
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0x8049031 <_start+49> mov ebx,0x0

native process 140785 In: _start L9 PC:
breakpoint already hit 1 time
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
(gdb) r i
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/avluckaya/work/arch-pc/lab09/lab09-2 i

Breakpoint 1, _start () at lab09-2.asm:9
(gdb)

```

Рис. 4.13: Режим псевдографики

4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился. Устанавливаю еще одну точку останова по адресу инструкции (рис. -fig. 4.14).


```
avluckaya@vbox:~/work/arch-pc/lab09$ gdb lab09-2

--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd0c0  0xffffd0c0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000  <_start>  eflags     0x202     [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

0x804aff7 add BYTE PTR [eax],al
0x804aff9 add BYTE PTR [eax],al
0x804affb add BYTE PTR [eax],al
0x804affd add BYTE PTR [eax],al
0x804afff .byte 0

native process 141528 In: _start L9 PC:
(gdb) r i
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/avluckaya/work/arch-pc/lab09/lab09-2 i

Breakpoint 1, _start () at lab09-2.asm:9
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num    Type             Disp Enb Address      What
1      breakpoint        keep y  0x08049000 lab09-2.asm:9
       breakpoint already hit 1 time
2      breakpoint        keep y  0x08049031 lab09-2.asm:20
(gdb)
```

Рис. 4.14: Добавление второй точки останова

4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой `info registers` (рис. -fig. 4.15).

```
aviuchkaya@vbox:~/work/arch-pc/lab09—gdb lab09-2
Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd0c0      0xffffd0c0      ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016      0x8049016 <_start+2>      eflags      0x202      43 [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>     mov     ebx,0x0

native process 141956 In: _start      L14      PC:
ebx      0x1      1
esp      0xffffd0c0      0xffffd0c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016      0x8049016 <_start+22>
eflags      0x202      43 [ 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--
fs       0x0      0
gs       0x0      0
(gdb)
```

Рис. 4.15: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. -fig. 4.16).

```

aviuchkaya@vbox:~/work/arch-pc/lab09—gdb lab09-2
Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd0c0      0xffffd0c0      ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016      0x8049016 <_start+2>      eflags      0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>     mov     ebx,0x0

native process 141956 In: _start      L14      PC:
edi      0x0      0
eip      0x8049016      0x8049016 <_start+22>
eflags   0x202      [ IF ]
cs       0x23      35
ss       0x2b      43
ds       0x2b      43
es       0x2b      43
fs       0x0      0
gs       0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb)

```

Рис. 4.16: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. -fig. 4.17).

```

aviukaya@vbox:~/work/arch-pc/lab09 - gdb lab09-2
Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd0c0      0xffffd0c0      ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016      0x8049016 <_start+2>      eflags      0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>     mov     ebx,0x0

native process 141956 In: _start      L14      PC:
fs       0x0      0
gs       0x0      0
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "xorld!\n\034"
(gdb)

```

Рис. 4.17: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. -fig. 4.18).

```

avdiukaya@vbox:~/work/arch-pc/lab09—gdb lab09-2
Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd0c0      0xffffd0c0      ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016      0x8049016 <_start+2>      eflags      0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>     mov     ebx,0x0

native process 141956 In: _start      L14      PC:
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "xorld!\n\034"
(gdb) p/t $ecx
$1 = 100000000100101000000000000000
(gdb) p/s $edx
$2 = 8
(gdb) p/t $edx
$3 = 1000
(gdb) p/x $edx
$4 = 0x8
(gdb)

```

Рис. 4.18: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. -fig. 4.19).

```

Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x2      2
esp      0xffffd0c0 0xffffd0c0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016 0x8049016 <_start+2  eflags    0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       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
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>     mov     ebx,0x0

native process 141956 In: _start
(gdb) p/s $edx
$2 = 8
(gdb) p/t $edx
$3 = 1000
(gdb) p/x $edx
$4 = 0x8
(gdb) set $ebx='2'
(gdb) p/s
$5 = 8
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb)

```

Рис. 4.19: Примеры использования команды set

4.1.4 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm. Создаю исполняемый файл. Загружаю исполняемый файл в отладчик, указав аргументы. (рис. -fig. 4.20).

```

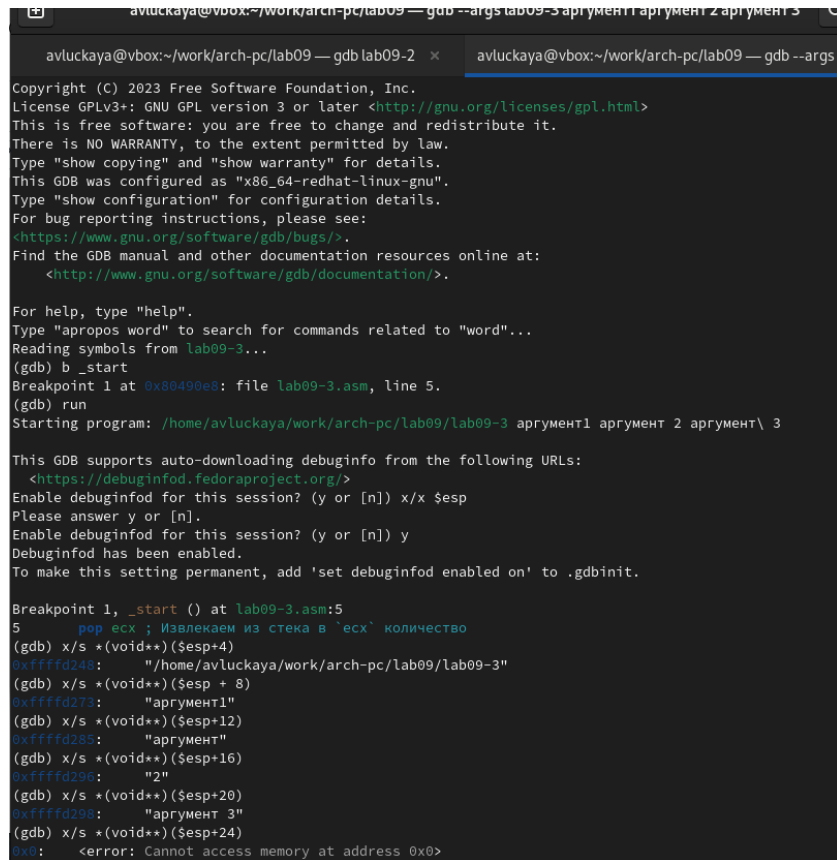
avluckaya@vbox:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
avluckaya@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
avluckaya@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
avluckaya@vbox:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Ubuntu 7.4.1-2ubuntu4) 7.4.1
Copyright (C) 2012 Free Software Foundation, Inc.
This is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
This program comes with ABSOLUTELY NO WARRANTY.
You may redistribute it under the terms of the GNU GPL, and
it is licensed to you under this same GPLv3 license.
Please see the file COPYING.
For more details about the GNU GPL, see the web site at
http://www.gnu.org/licenses/gpl.html.
GNU gdb (Ubuntu 7.4.1-2ubuntu4) 7.4.1
Copyright (C) 2012 Free Software Foundation, Inc.
This is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
This program comes with ABSOLUTELY NO WARRANTY.
You may redistribute it under the terms of the GNU GPL, and
it is licensed to you under this same GPLv3 license.
Please see the file COPYING.
For more details about the GNU GPL, see the web site at
http://www.gnu.org/licenses/gpl.html.
(gdb)

```

Рис. 4.20: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпоинт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы,

а указатель `void` занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились. (рис. -fig. 4.16).



```
avluckaya@vbox:~/work/arch-pc/lab09 — gdb --args lab09-3 аргумент1 аргумент2 аргумент3
avluckaya@vbox:~/work/arch-pc/lab09 — gdb lab09-2
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) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/avluckaya/work/arch-pc/lab09/lab09-3 аргумент1 аргумент2 аргумент\ 3

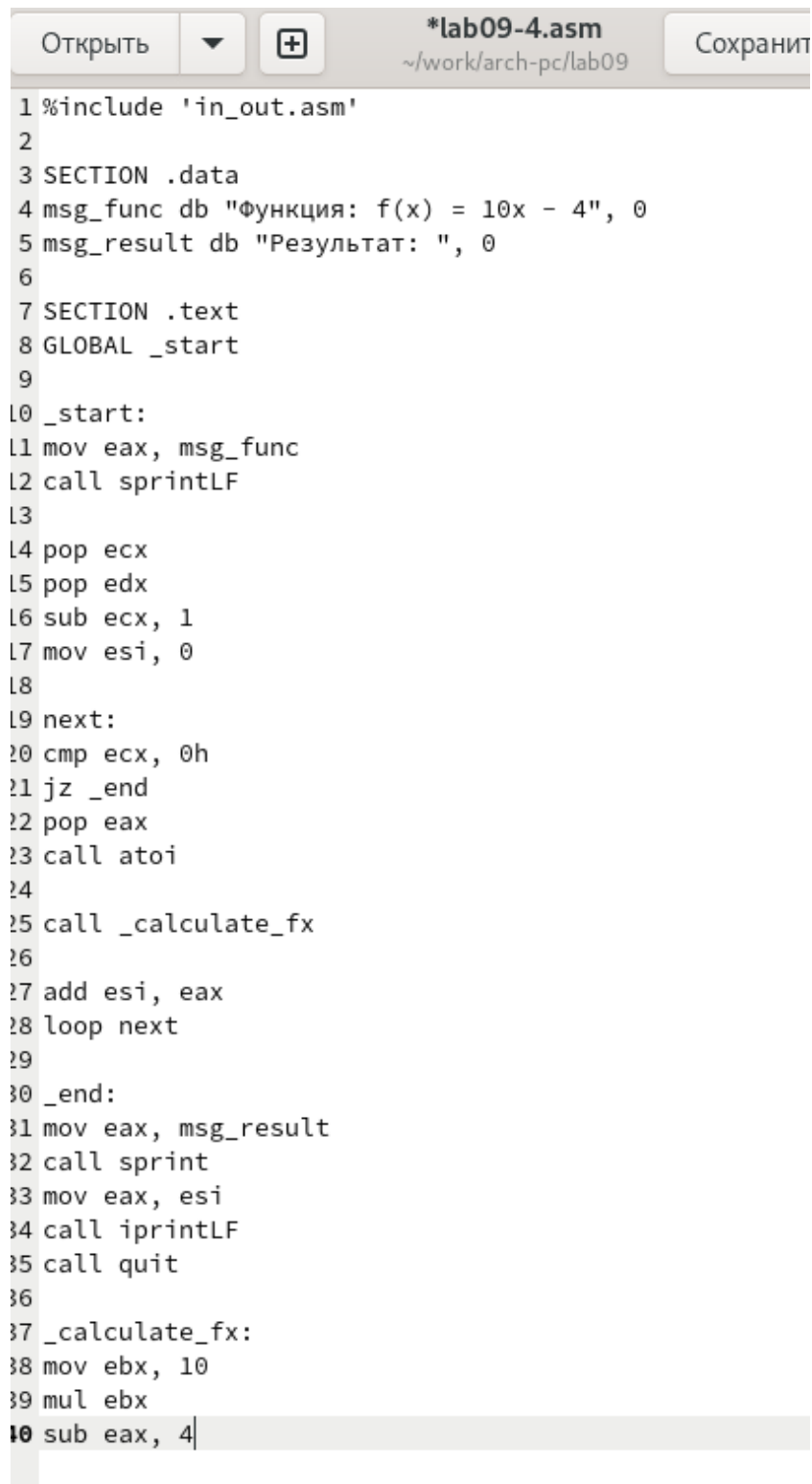
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) x/x $esp
Please answer y or [n].
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, 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) x/s *(void**)(esp+4)
0xffffd248: "/home/avluckaya/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd273: "аргумент1"
(gdb) x/s *(void**)(esp+12)
0xffffd285: "аргумент"
(gdb) x/s *(void**)(esp+16)
0xffffd296: "2"
(gdb) x/s *(void**)(esp+20)
0xffffd298: "аргумент 3"
(gdb) x/s *(void**)(esp+24)
0x0: <error: Cannot access memory at address 0x0>
```

Рис. 4.21: Проверка работы стека

4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. -fig. 4.22).



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

Рис. 4.22: Измененная программа предыдущей лабораторной работы

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию (рис. -fig. 4.23).

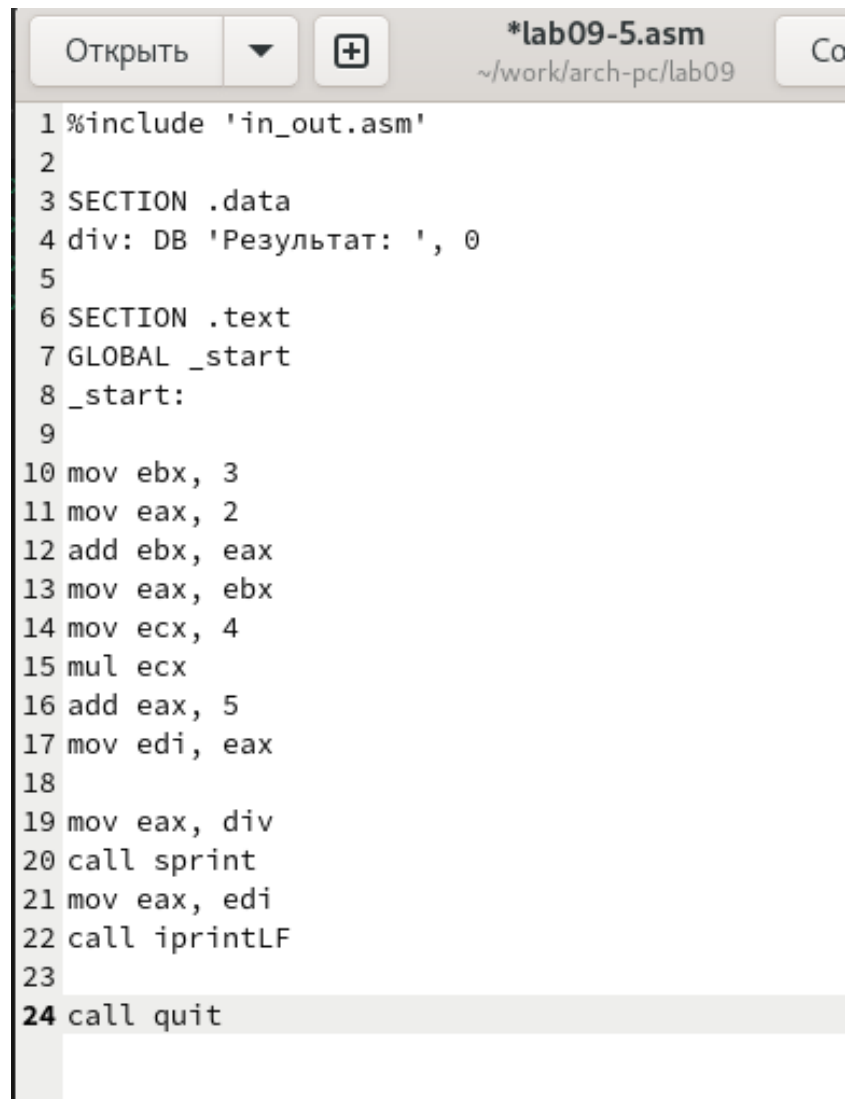
```
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0c0 0xffffd0c0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049014 0x8049014 <_start+20>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
b+ 0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80

native process 147434 In: _start L13 PC: 0
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0c0 0xffffd0c0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049014 0x8049014 <_start+20>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 4.23: Поиск ошибки в программе через пошаговую отладку

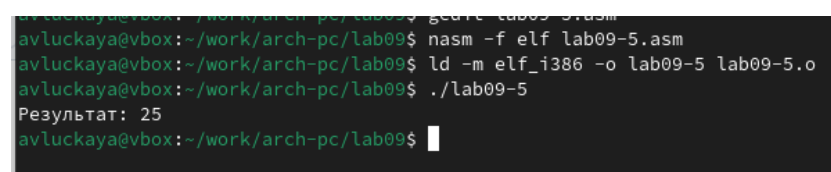
Исправляю найденную ошибку (рис. -fig. 4.24).



```
Открыть ▼ + *lab09-5.asm ~/work/arch-pc/lab09 Co
1 %include 'in_out.asm'
2
3 SECTION .data
4 div: DB 'Результат: ', 0
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov ebx, 3
11 mov eax, 2
12 add ebx, eax
13 mov eax, ebx
14 mov ecx, 4
15 mul ecx
16 add eax, 5
17 mov edi, eax
18
19 mov eax, div
20 call sprint
21 mov eax, edi
22 call iprintLF
23
24 call quit
```

Рис. 4.24: Исправление ошибки

Проверка выполнения программы (рис. -fig. 4.25).



```
avluckaya@vbox: ~/work/arch-pc/lab09$ gcc -c lab09-5.asm
avluckaya@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
avluckaya@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
avluckaya@vbox:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
avluckaya@vbox:~/work/arch-pc/lab09$
```

Рис. 4.25: Проверка корректировок в программе

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

6 Список литературы

1. <https://esystem.rudn.ru/course/view.php?id=112>

...