

# **Отчёт по лабораторной работе 9**

**Архитектура компьютеров**

Хаммудех Салех

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация подпрограмм в NASM . . . . .	6
2.2	Отладка программы с помощью GDB . . . . .	10
2.3	Задание для самостоятельной работы . . . . .	20
<b>3</b>	<b>Выводы</b>	<b>27</b>

## Список иллюстраций

2.1	Текст программы lab9-1.asm . . . . .	7
2.2	Запуск программы lab9-1.asm . . . . .	8
2.3	Модифицированная программа lab9-1.asm . . . . .	9
2.4	Запуск модифицированной программы lab9-1.asm . . . . .	9
2.5	Код программы lab9-2.asm . . . . .	10
2.6	Запуск программы lab9-2.asm в GDB . . . . .	11
2.7	Дизассемблированный код программы . . . . .	12
2.8	Дизассемблированный код в Intel-синтаксисе . . . . .	13
2.9	Настройка точки останова . . . . .	14
2.10	Отслеживание изменений регистров . . . . .	15
2.11	Детальный анализ регистров . . . . .	16
2.12	Изменение значения переменной msg1 . . . . .	17
2.13	Просмотр регистра после изменений . . . . .	18
2.14	Изменение значения регистра . . . . .	19
2.15	Анализ стека программы . . . . .	20
2.16	Код программы lab9-prog.asm . . . . .	21
2.17	Запуск программы lab9-prog.asm . . . . .	22
2.18	Код с ошибкой . . . . .	23
2.19	Процесс отладки программы . . . . .	24
2.20	Исправленный код программы . . . . .	25
2.21	Проверка исправленного кода . . . . .	26

## **Список таблиц**

# 1 Цель работы

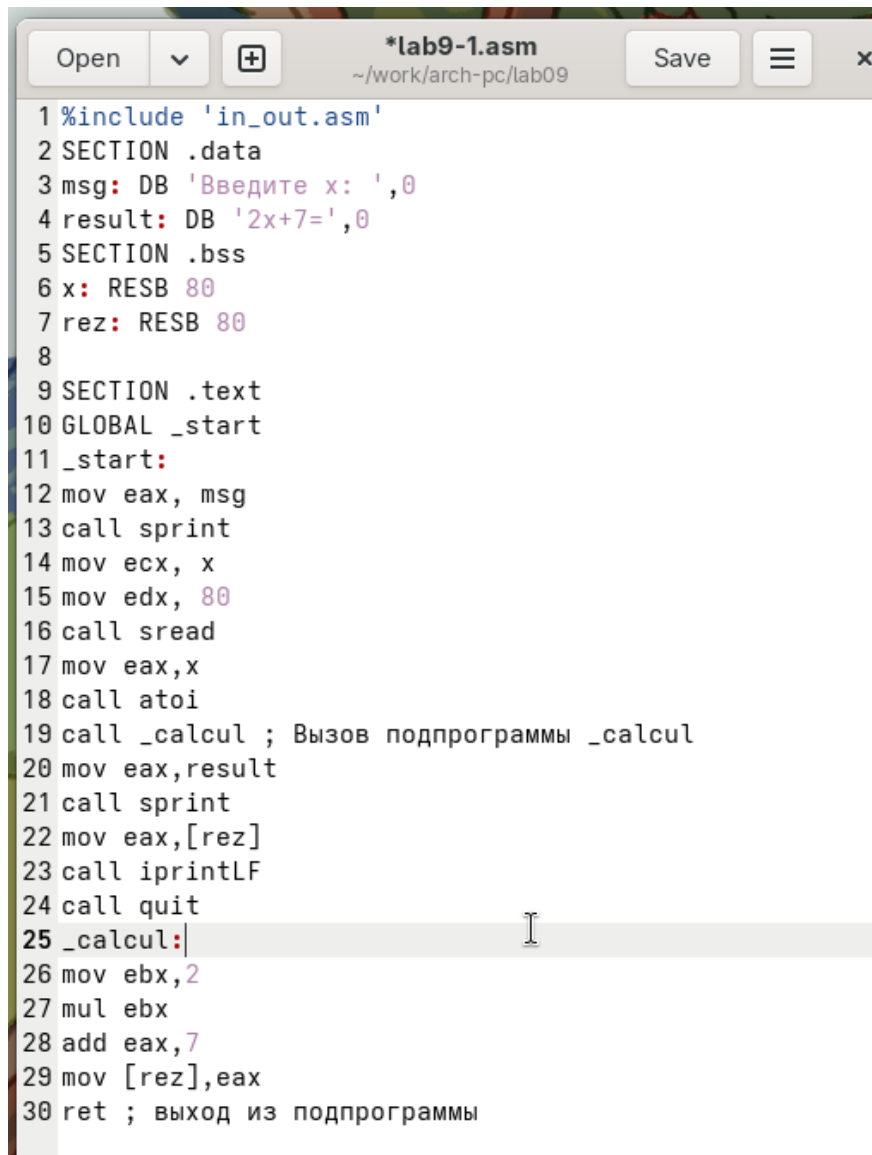
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Выполнение лабораторной работы

### 2.1 Реализация подпрограмм в NASM

Для выполнения лабораторной работы №9 я создал новую папку и перешел в нее. Затем я создал файл с именем lab9-1.asm.

В качестве примера была рассмотрена программа, которая вычисляет арифметическое выражение  $f(x) = 2x + 7$  с использованием подпрограммы `calcul`. Значение переменной  $x$  вводится с клавиатуры, а вычисление производится внутри подпрограммы. (рис. 2.1) (рис. 2.2)



```
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 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
```

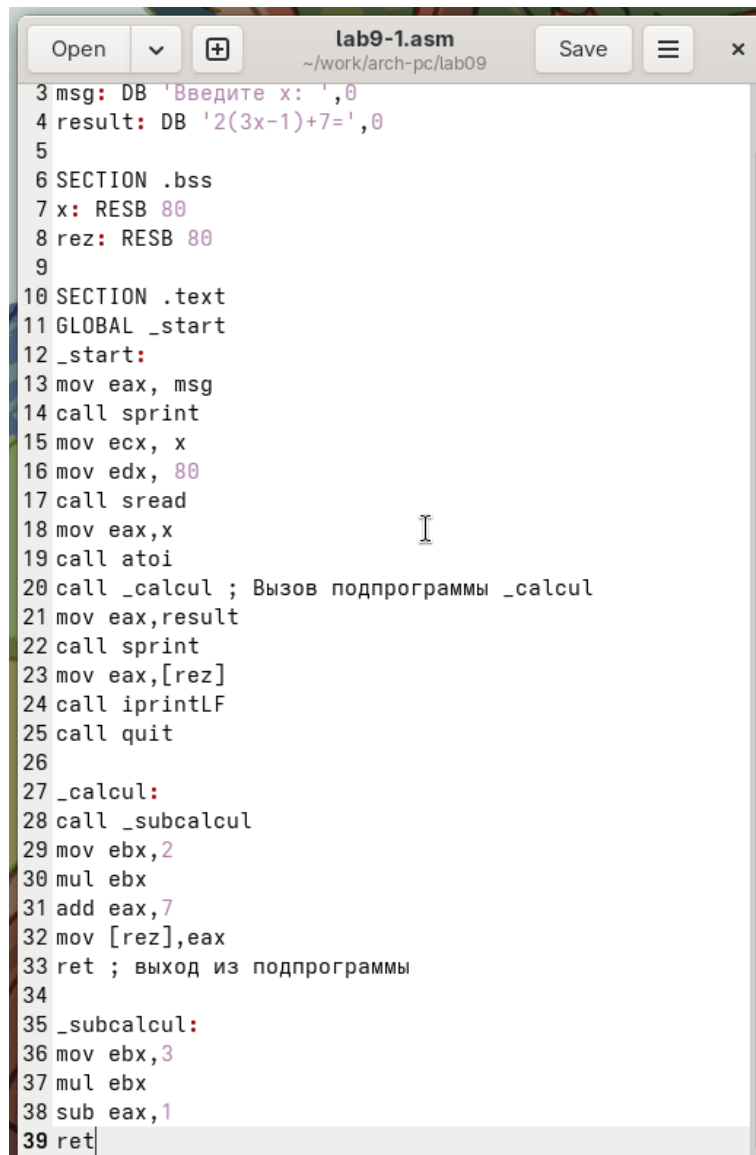
Рисунок 2.1: Текст программы lab9-1.asm

```
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
lab9-1.asm:1: error: unable to open include file `in_out.asm': No such file or d
irectory
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2x+7=21
hammudehsaleh@fedora:~/work/arch-pc/lab09$
```

Рисунок 2.2: Запуск программы lab9-1.asm

Далее я модифицировал программу, добавив подпрограмму subcalcul внутри подпрограммы calcul. Это позволило вычислять составное выражение  $f(g(x))$ , где  $f(x) = 2x + 7$ , а  $g(x) = 3x - 1$ . Значение  $x$  вводится с клавиатуры. (рис. 2.3) (рис. 2.4)





```
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
```

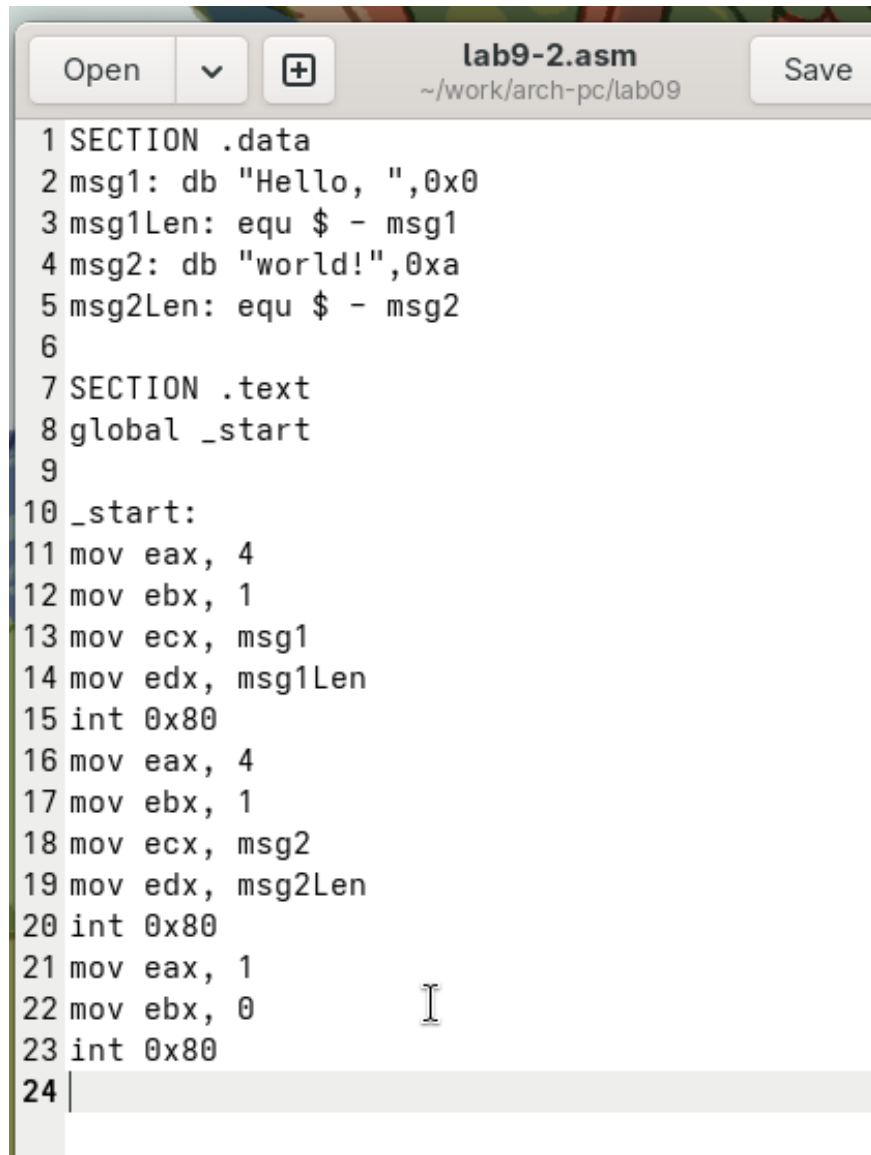
Рисунок 2.3: Модифицированная программа lab9-1.asm

```
hammudehsaleh@fedora:~/work/arch-pc/lab09$
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2(3x-1)+7=47
hammudehsaleh@fedora:~/work/arch-pc/lab09$
```

Рисунок 2.4: Запуск модифицированной программы lab9-1.asm

## 2.2 Отладка программы с помощью GDB

Я создал файл lab9-2.asm, в котором содержится программа из Листинга 9.2. Она отвечает за вывод сообщения «Hello world!» на экран. (рис. 2.5)



```
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
24
```

Рисунок 2.5: Код программы lab9-2.asm

После компиляции с ключом -g для добавления отладочной информации я загрузил исполняемый файл в GDB. Запустил программу с помощью команды run или r. (рис. 2.6)

```

hammudehsaleh@fedora:~/work/arch-pc/lab09$
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
hammudehsaleh@fedora:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 16.2-3.fc42
Copyright (C) 2024 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/hammudehsaleh/work/arch-pc/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 12936) exited normally]
(gdb)

```

Рисунок 2.6: Запуск программы lab9-2.asm в GDB

Для анализа программы я установил точку остановки на метке `_start` и запустил выполнение. Затем изучил дизассемблированный код программы. (рис. 2.7) (рис. 2.8)

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Starting program: /home/hammudehsaleh/work/arch-pc/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 12936) exited normally]
(gdb) break _start
Breakpoint 1 at 0x08048080: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/hammudehsaleh/work/arch-pc/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:
=> 0x08048080 <+0>:      mov     $0x4,%eax
0x08048085 <+5>:      mov     $0x1,%ebx
0x0804808a <+10>:     mov     $0x8049000,%ecx
0x0804808f <+15>:     mov     $0x8,%edx
0x08048094 <+20>:     int     $0x80
0x08048096 <+22>:     mov     $0x4,%eax
0x0804809b <+27>:     mov     $0x1,%ebx
0x080480a0 <+32>:     mov     $0x8049008,%ecx
0x080480a5 <+37>:     mov     $0x7,%edx
0x080480aa <+42>:     int     $0x80
0x080480ac <+44>:     mov     $0x1,%eax
0x080480b1 <+49>:     mov     $0x0,%ebx
0x080480b6 <+54>:     int     $0x80
End of assembler dump.
(gdb) 
```

Рисунок 2.7: Дизассемблированный код программы

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Dump of assembler code for function _start:
=> 0x08048080 <+0>:      mov     $0x4,%eax
    0x08048085 <+5>:      mov     $0x1,%ebx
    0x0804808a <+10>:     mov     $0x8049000,%ecx
    0x0804808f <+15>:     mov     $0x8,%edx
    0x08048094 <+20>:     int     $0x80
    0x08048096 <+22>:     mov     $0x4,%eax
    0x0804809b <+27>:     mov     $0x1,%ebx
    0x080480a0 <+32>:     mov     $0x8049008,%ecx
    0x080480a5 <+37>:     mov     $0x7,%edx
    0x080480aa <+42>:     int     $0x80
    0x080480ac <+44>:     mov     $0x1,%eax
    0x080480b1 <+49>:     mov     $0x0,%ebx
    0x080480b6 <+54>:     int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:      mov     eax,0x4
    0x08048085 <+5>:      mov     ebx,0x1
    0x0804808a <+10>:     mov     ecx,0x8049000
    0x0804808f <+15>:     mov     edx,0x8
    0x08048094 <+20>:     int     0x80
    0x08048096 <+22>:     mov     eax,0x4
    0x0804809b <+27>:     mov     ebx,0x1
    0x080480a0 <+32>:     mov     ecx,0x8049008
    0x080480a5 <+37>:     mov     edx,0x7
    0x080480aa <+42>:     int     0x80
    0x080480ac <+44>:     mov     eax,0x1
    0x080480b1 <+49>:     mov     ebx,0x0
    0x080480b6 <+54>:     int     0x80
End of assembler dump.
(gdb) 
```

Рисунок 2.8: Дизассемблированный код в Intel-синтаксисе

Для проверки точки останова я использовал команду `info breakpoints (i b)`. Установил дополнительную точку останова по адресу инструкции `mov ebx, 0x0`. (рис. 2.9)

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffce50 0xffffce50
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048080 0x8048080 <_start>

B+>0x8048080 <_start> mov    eax,0x4
0x8048085 <_start+5> mov    ebx,0x1
0x804808a <_start+10> mov    ecx,0x8049000
0x804808f <_start+15> mov    edx,0x8
0x8048094 <_start+20> int     0x80
0x8048096 <_start+22> mov    eax,0x4
0x804809b <_start+27> mov    ebx,0x1
0x80480a0 <_start+32> mov    ecx,0x8049008
0x80480a5 <_start+37> mov    edx,0x7

native process 12942 (asm) In: _start L11 PC: 0x8048080
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031
(gdb) i b
Num    Type      Disp Enb Address      What
1      breakpoint keep y 0x08048080 lab9-2.asm:11
       breakpoint already hit 1 time
2      breakpoint keep y 0x08049031
(gdb) |
```

Рисунок 2.9: Настройка точки останова

С помощью команды `stepi (si)` выполнил пошаговую отладку, отслеживая изменения регистров. (рис. 2.10) (рис. 2.11)

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffce50 0xffffce50
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048085 0x8048085 <_start+5>

B+ 0x8048080 <_start> mov eax,0x4
>0x8048085 <_start+5> mov ebx,0x1
0x804808a <_start+10> mov ecx,0x8049000
0x804808f <_start+15> mov edx,0x8
0x8048094 <_start+20> int 0x80
0x8048096 <_start+22> mov eax,0x4
0x804809b <_start+27> mov ebx,0x1
0x80480a0 <_start+32> mov ecx,0x8049008
0x80480a5 <_start+37> mov edx,0x7

native process 12942 (asm) In: _start L12 PC: 0x8048085
eip      0x8048080 0x8048080 <_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) 
```

Рисунок 2.10: Отслеживание изменений регистров

The screenshot shows a GDB debugger window titled "hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into several sections. At the top, there's a "Register group: general" section showing the values of general-purpose registers: eax (0x8), ecx (0x8049000), edx (0x8), ebx (0x1), esp (0xffffce50), ebp (0x0), esi (0x0), edi (0x0), and eip (0x8048096). Below this, there's a disassembly window showing the assembly code for the current instruction set. The code starts at address 0x8048080 and ends at 0x80480a5. The instructions are: mov eax, 0x4; mov ebx, 0x1; mov ecx, 0x8049000; mov edx, 0x8; int 0x80; mov eax, 0x4; mov ebx, 0x1; mov ecx, 0x8049008; mov edx, 0x7. The instruction at 0x8048096 is highlighted. At the bottom, there's a "native process 12942 (asm) In: \_start" section showing the values of segment registers: es (0x2b), fs (0x0), and gs (0x0). The PC is 0x8048096. The GDB prompt shows the user has entered "si" and "is", and the debugger has responded with "Undefined command: 'is'. Try 'help'".

```
Register group: general
eax      0x8      8
ecx      0x8049000 134516736
edx      0x8      8
ebx      0x1      1
esp      0xffffce50 0xffffce50
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048096 0x8048096 <_start+22>

B+ 0x8048080 <_start>      mov     eax, 0x4
0x8048085 <_start+5>      mov     ebx, 0x1
0x804808a <_start+10>     mov     ecx, 0x8049000
0x804808f <_start+15>     mov     edx, 0x8
0x8048094 <_start+20>     int     0x80
>0x8048096 <_start+22>     mov     eax, 0x4
0x804809b <_start+27>     mov     ebx, 0x1
0x80480a0 <_start+32>     mov     ecx, 0x8049008
0x80480a5 <_start+37>     mov     edx, 0x7

native process 12942 (asm) In: _start      L16      PC: 0x8048096
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) is
Undefined command: "is". Try "help".
(gdb) si
(gdb) si
(gdb) siHello,
(gdb)
```

Рисунок 2.11: Детальный анализ регистров

Я также просмотрел значение переменной msg1 по имени и изменил первый символ переменной с помощью команды set. (рис. 2.12) (рис. 2.13)



```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Register group: general
eax      0x8      8
ecx      0x8049000 134516736
edx      0x8      8
ebx      0x1      1
esp      0xffffce50 0xffffce50
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048096 0x8048096 <_start+22>

B+ 0x8048080 <_start>      mov     eax,0x4
0x8048085 <_start+5>      mov     ebx,0x1
0x804808a <_start+10>     mov     ecx,0x8049000
0x804808f <_start+15>     mov     edx,0x8
0x8048094 <_start+20>     int     0x80
>0x8048096 <_start+22>     mov     eax,0x4
0x804809b <_start+27>     mov     ebx,0x1
0x80480a0 <_start+32>     mov     ecx,0x8049008
0x80480a5 <_start+37>     mov     edx,0x7

native process 12942 (asm) In: _start L16 PC: 0x8048096
0x8049000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008:      <error: Cannot access memory at address 0x804a008>
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x8049000 <msg1>:      "hello, "
(gdb) set {char}0x804a008='L'
Cannot access memory at address 0x804a008
(gdb) x/1sb 0x804a008
0x804a008:      <error: Cannot access memory at address 0x804a008>
(gdb) 
```

Рисунок 2.12: Изменение значения переменной msg1

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Register group: general
eax      0x8      8
ecx      0x8049000 134516736
edx      0x8      8
ebx      0x1      1
esp      0xffffce50 0xffffce50
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048096 0x8048096 <_start+22>

B+ 0x8048080 <_start>      mov     eax,0x4
0x8048085 <_start+5>      mov     ebx,0x1
0x804808a <_start+10>     mov     ecx,0x8049000
0x804808f <_start+15>     mov     edx,0x8
0x8048094 <_start+20>     int     0x80
>0x8048096 <_start+22>     mov     eax,0x4
0x804809b <_start+27>     mov     ebx,0x1
0x80480a0 <_start+32>     mov     ecx,0x8049008
0x80480a5 <_start+37>     mov     edx,0x7

native process 12942 (asm) In: _start L16 PC: 0x8048096
(gdb) p/s $ecx
$3 = 134516736
(gdb) p/x $ecx
$4 = 0x8049000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) █
```

Рисунок 2.13: Просмотр регистра после изменений

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Register group: general
eax      0x8      8
ecx      0x8049000 134516736
edx      0x8      8
ebx      0x2      2
esp      0xffffce50 0xffffce50
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8048096 0x8048096 <_start+22>

B+ 0x8048080 <_start>    mov    eax,0x4
0x8048085 <_start+5>    mov    ebx,0x1
0x804808a <_start+10>   mov    ecx,0x8049000
0x804808f <_start+15>   mov    edx,0x8
0x8048094 <_start+20>   int     0x80
>0x8048096 <_start+22>   mov    eax,0x4
0x804809b <_start+27>   mov    ebx,0x1
0x80480a0 <_start+32>   mov    ecx,0x8049008
0x80480a5 <_start+37>   mov    edx,0x7

native process 12942 (asm) In: _start L16 PC: 0x8048096
(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) 
```

Рисунок 2.14: Изменение значения регистра

Для проверки программы с аргументами я скопировал файл lab8-2.asm из лабораторной работы №8, создал исполняемый файл и загрузил его в GDB с помощью ключа `-args`. Затем исследовал стек, где хранились адреса аргументов. (рис. 2.15)

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 argu...
~/work/arch-pc/lab09

<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x8048148: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/hammudehsaleh/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argume

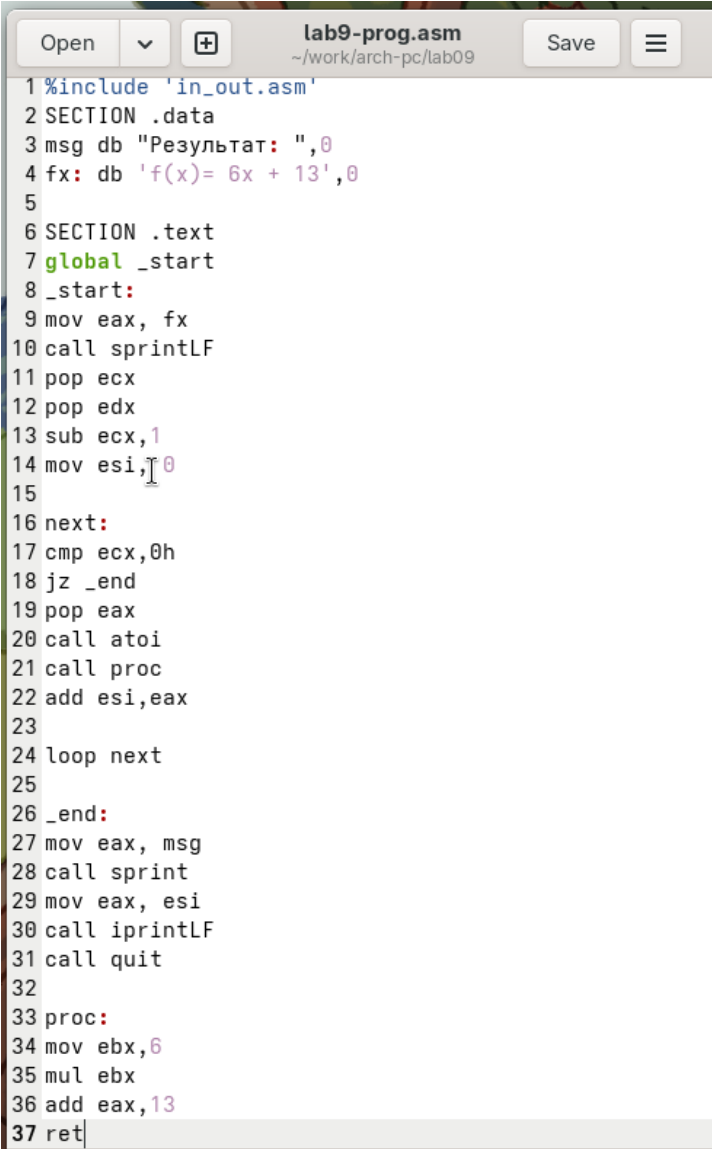
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffce20: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd000: "/home/hammudehsaleh/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd02e: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd037: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd039: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd042: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd044: "argument 3"
(gdb) █
```

Рисунок 2.15: Анализ стека программы

## 2.3 Задание для самостоятельной работы

Я модифицировал программу из лабораторной работы №8, добавив вычисление функции  $f(x)$  в виде подпрограммы. (рис. 2.16) (рис. 2.17)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 6x + 13',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintf
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 proc
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprintf
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 proc:
34 mov ebx,6
35 mul ebx
36 add eax,13
37 ret
```

Рисунок 2.16: Код программы lab9-prog.asm

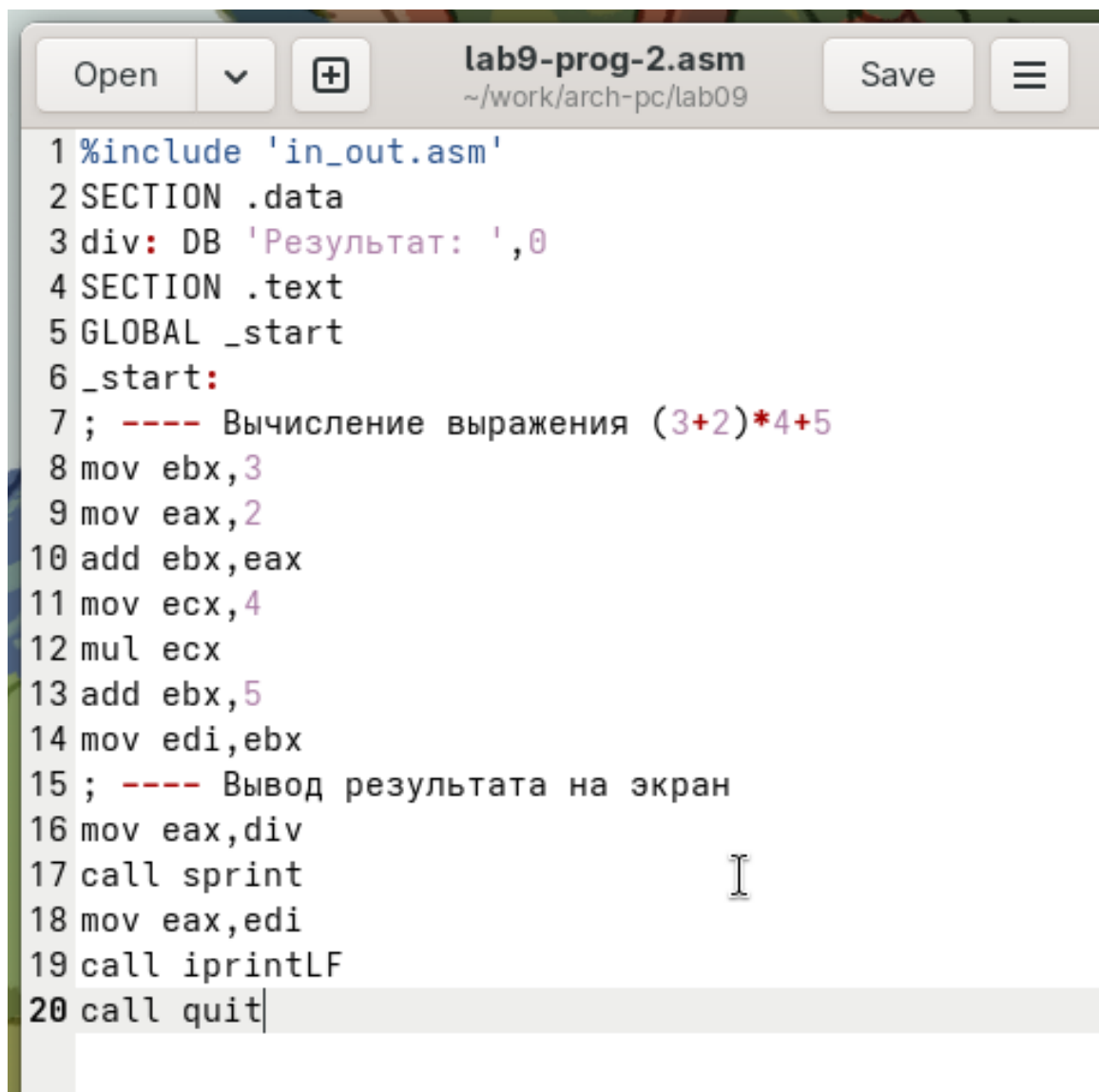
```

hammudehsaleh@fedora:~/work/arch-pc/lab09$
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-prog.asm
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-prog.o -o lab9-prog
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ./lab9-prog
f(x)= 6x + 13
Результат: 0
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ./lab9-prog 3
f(x)= 6x + 13
Результат: 31
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ./lab9-prog 3 4 5 6
f(x)= 6x + 13
Результат: 160
hammudehsaleh@fedora:~/work/arch-pc/lab09$

```

Рисунок 2.17: Запуск программы lab9-prog.asm

При запуске программы я обнаружил ошибку: результат вычислений был неверным. Анализ с помощью GDB показал, что аргументы инструкции add перепутаны, а по окончании программы значение регистра ebx вместо eax отправляется в edi. (рис. 2.18) (рис. 2.19)

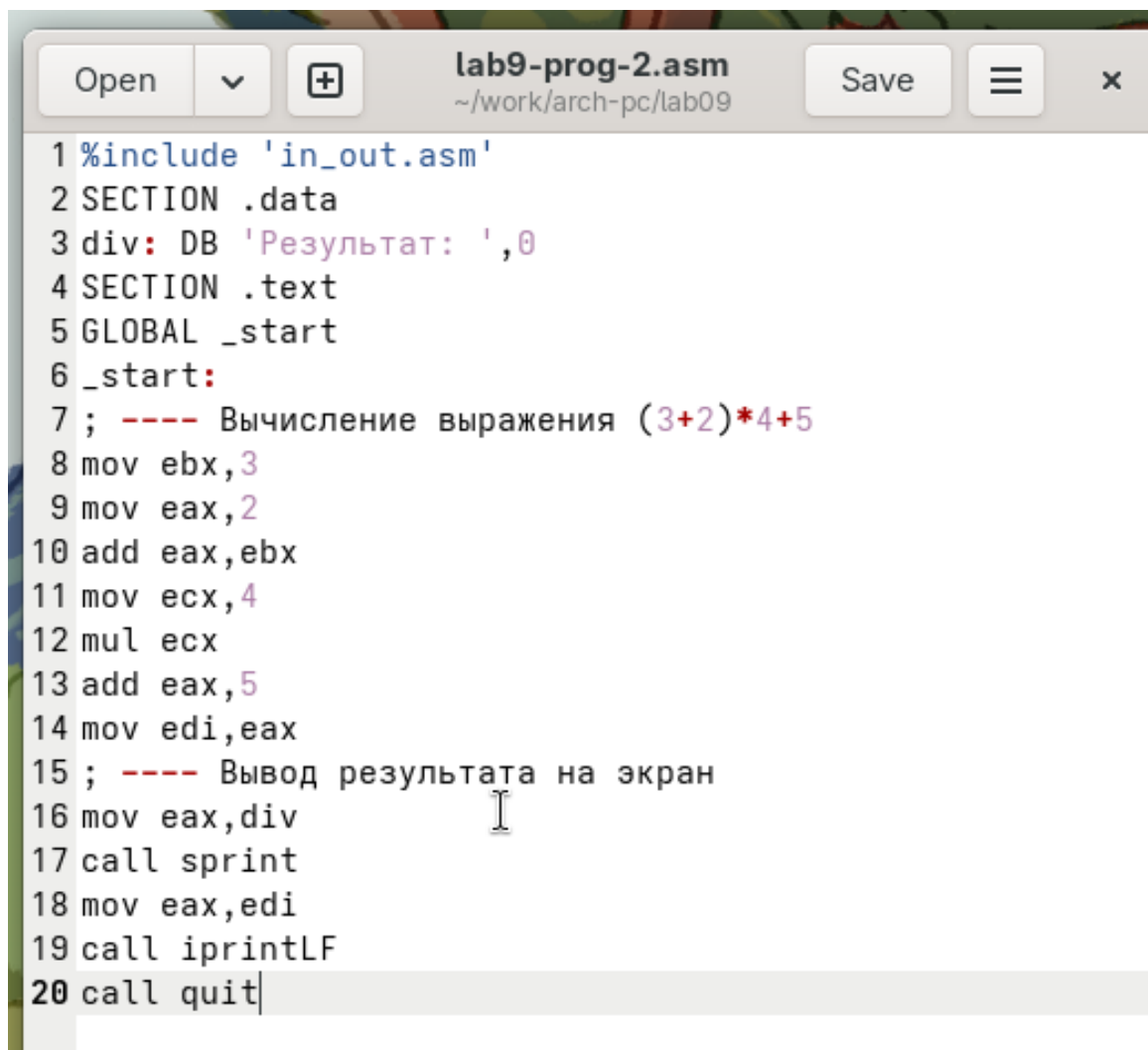


```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ---- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add ebx,eax
11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
```

Рисунок 2.18: Код с ошибкой







The image shows a code editor window titled 'lab9-prog-2.asm' with the path '~/.work/arch-pc/lab09'. The editor contains 20 lines of assembly code. The code defines a data section with a string 'Результат: ',0 and a text section starting at \_start. It calculates the expression (3+2)\*4+5 using registers ebx, eax, ecx, and edi. The result is then printed using the 'div' variable and 'sprint' function, followed by 'iprintLF' and 'quit'.

```
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
```

Рисунок 2.20: Исправленный код программы

```
hammudehsaleh@fedora:~/work/arch-pc/lab09 — gdb lab9-prog-2
~/work/arch-pc/lab09

eax      s----- 25

      fffce40      xffffce40
      [ Register Values Unavailable ]

      9      5

0x8048180 <_start+24>  mul    eax,0x8049000
0x804818a <_start+34>  mov    eax,edi
0x804818c <_start+36>  call  0x8048106 <iprintLF>
0x8048191 <_start+41>  call  0x804815b <quit>

native process 13232 (asm) In: _start L16 PC: 0x8048180
To makeNo process (asm) In: set debuginfod enabled off' to .gdbinit. L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cРезультат: 25

Continuing.
[Inferior 1 (process 13232) exited normally]
(gdb) 
```

Рисунок 2.21: Проверка исправленного кода

## 3 Выводы

Я освоил работу с подпрограммами и отладчиком GDB, научился находить и исправлять ошибки в коде с помощью анализа стеков, регистров и дизассемблированного кода.