

Отчёт по лабораторной работе 9

дисциплина: Архитектура компьютера

Шангина В. А НКАбд-05-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программы с помощью GDB	10
2.3	Работа с аргументами командной строки	20
2.4	Задание для самостоятельной работы	21
2.5	Выводы	26

Список иллюстраций

2.1	Исходный код программы lab9-1.asm	7
2.2	Результат выполнения программы	8
2.3	Модифицированный код программы	9
2.4	Результат выполнения модифицированной программы	10
2.5	Код программы lab9-2.asm	11
2.6	Запуск программы в отладчике	12
2.7	Дизассемблированный код	13
2.8	Дизассемблированный код в режиме Intel	14
2.9	Установка точки останова	15
2.10	Изменение значений регистров	16
2.11	Отслеживание изменений регистров	17
2.12	Изменение переменной	18
2.13	Отображение измененного регистра	19
2.14	Изменение регистра ebx	20
2.15	Просмотр аргументов командной строки	21
2.16	Код программы prog-1.asm	22
2.17	Результат выполнения программы	22
2.18	Код с ошибками	23
2.19	Результат отладки	24
2.20	Исправленный код программы	25
2.21	Результат проверки	25

Список таблиц

1 Цель работы

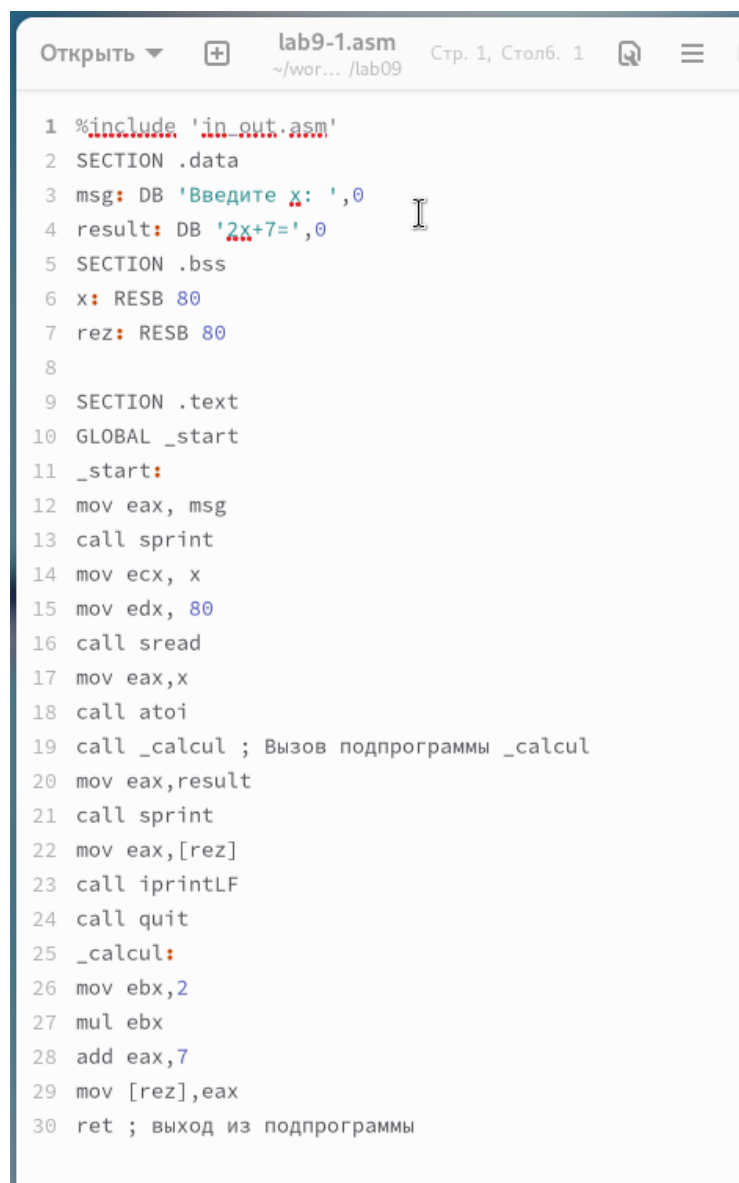
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

Сначала я создала новую папку для выполнения лабораторной работы №9 и перешла в нее. Затем создала файл с именем lab9-1.asm.

В качестве примера я реализовала программу, вычисляющую арифметическое выражение $f(x) = 2x + 7$ с использованием подпрограммы calcul. Значение переменной x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.



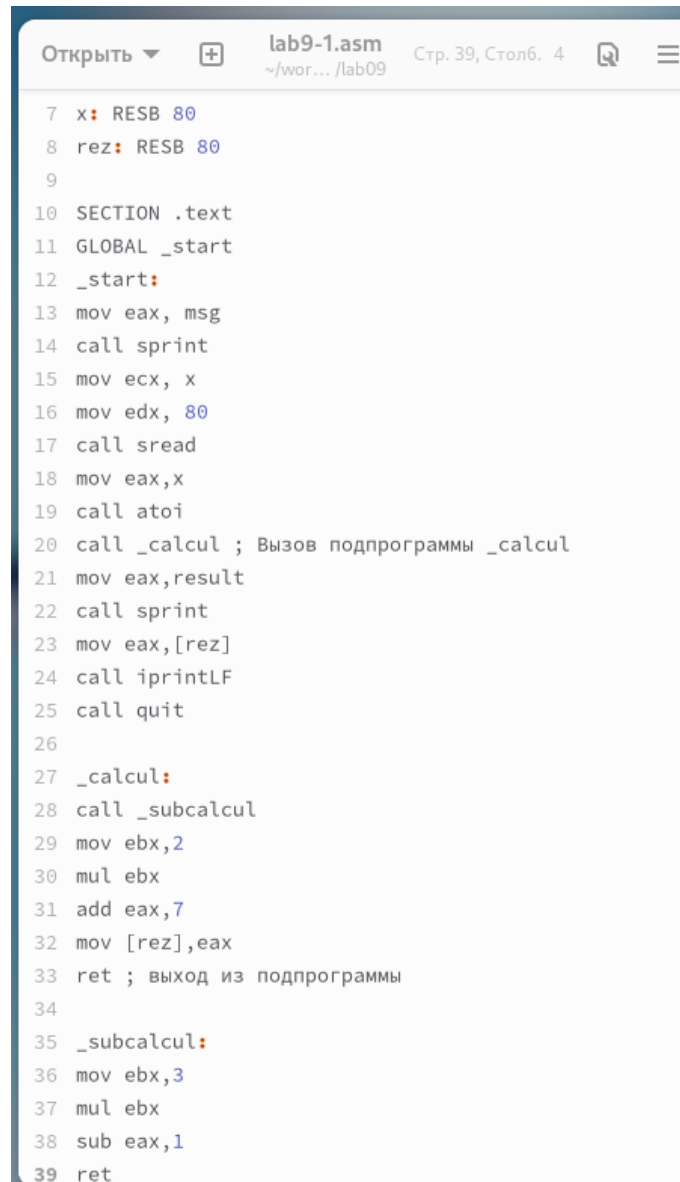
```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 rez: RESB 80
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax,x
18 call atoi
19 call _calcul ; Вызов подпрограммы _calcul
20 mov eax,result
21 call sprint
22 mov eax,[rez]
23 call iprintLF
24 call quit
25 _calcul:
26 mov ebx,2
27 mul ebx
28 add eax,7
29 mov [rez],eax
30 ret ; выход из подпрограммы
```

Рис. 2.1: Исходный код программы lab9-1.asm

```
victoriashangina@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
victoriashangina@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
victoriashangina@vbox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2x+7=21
victoriashangina@vbox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 2
2x+7=11
victoriashangina@vbox:~/work/arch-pc/lab09$
```

Рис. 2.2: Результат выполнения программы

Затем я изменила текст программы, добавив подпрограмму `subcalcul` внутрь подпрограммы `calcul`. Это позволило вычислять составное выражение $f(g(x))$, где $f(x) = 2x + 7$, $g(x) = 3x - 1$. Значение x также вводится с клавиатуры.



```
7 x: RESB 80
8 rez: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x
19 call atoi
20 call _calcul ; Вызов подпрограммы _calcul
21 mov eax, result
22 call sprint
23 mov eax, [rez]
24 call iprintLF
25 call quit
26
27 _calcul:
28 call _subcalcul
29 mov ebx, 2
30 mul ebx
31 add eax, 7
32 mov [rez], eax
33 ret ; выход из подпрограммы
34
35 _subcalcul:
36 mov ebx, 3
37 mul ebx
38 sub eax, 1
39 ret
```

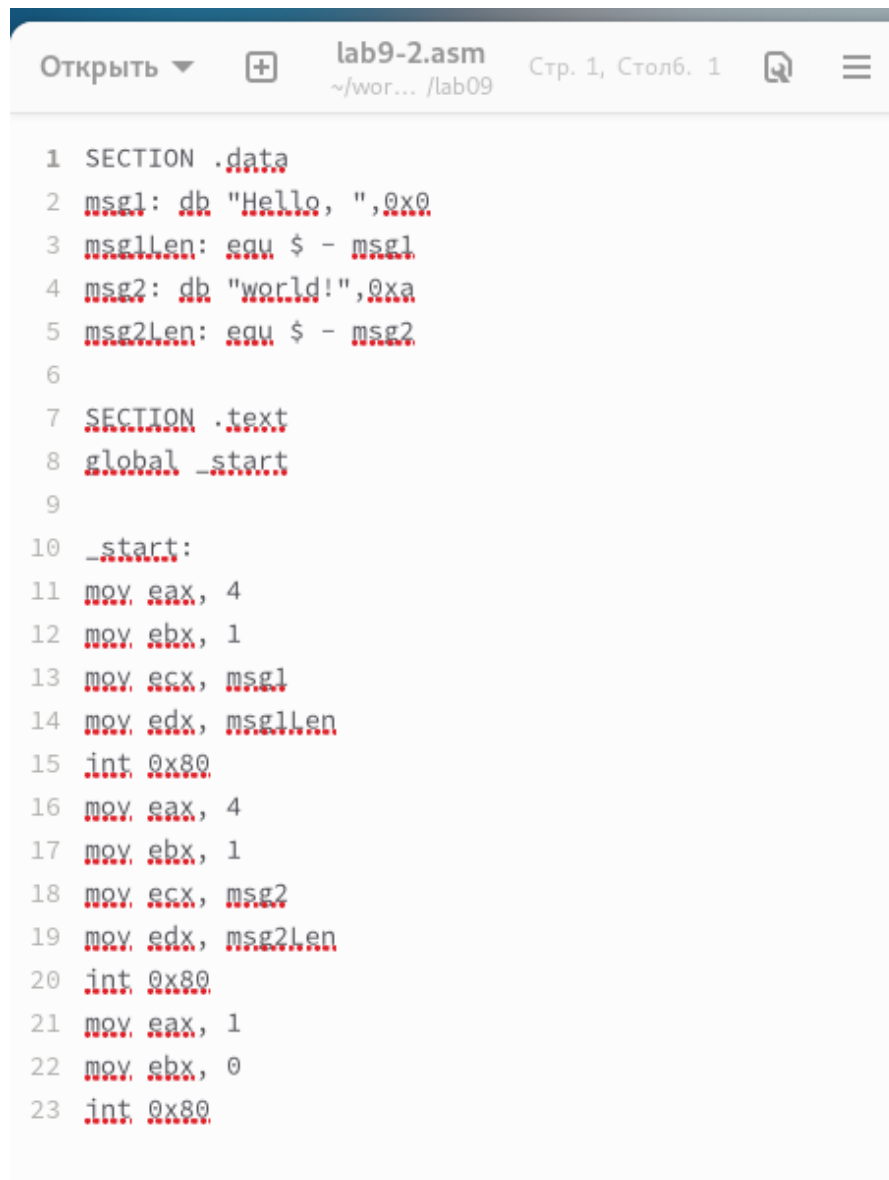
Рис. 2.3: Модифицированный код программы

```
victoriashangina@vbox:~/work/arch-pc/lab09$  
victoriashangina@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm  
victoriashangina@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o  
victoriashangina@vbox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 7  
2(3x-1)+7=47  
victoriashangina@vbox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 2  
2(3x-1)+7=17  
victoriashangina@vbox:~/work/arch-pc/lab09$
```

Рис. 2.4: Результат выполнения модифицированной программы

2.2 Отладка программы с помощью GDB

Я создала файл lab9-2.asm, содержащий программу для вывода сообщения “Hello, world!” (Листинг 9.2).



```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2len: equ $ - msg2
6
7 SECTION .text
8 global _start
9
10 _start:
11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msg1len
15 int 0x80
16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2len
20 int 0x80
21 mov eax, 1
22 mov ebx, 0
23 int 0x80
```

Рис. 2.5: Код программы lab9-2.asm

Скомпилировала файл с ключом -g для добавления отладочной информации и загрузила его в GDB. Затем запустила программу командой run.

```

victoriashangina@vbox:~/work/arch-pc/lab09$
victoriashangina@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
victoriashangina@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
victoriashangina@vbox:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.1-1.fc39
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/victoriashangina/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 3709) exited normally]
(gdb)

```

Рис. 2.6: Запуск программы в отладчике

Установила точку останова на метке `_start`, запустила программу, а затем просмотрела дизассемблированный код.

```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb 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 3709) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/victoriashangina/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:
=> 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. 2.7: Дизассемблированный код

```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb lab9-2
=> 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. 2.8: Дизассемблированный код в режиме Intel

Установила дополнительные точки останова, используя команды `info breakpoints` и `break`. Например, добавила точку на инструкции `mov ebx, 0x0`.

```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0b0 0xffffd0b0
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 3715 (asm) In: _start L11 PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y  0x08049000 lab9-2.asm:11
       breakpoint already hit 1 time
2      breakpoint      keep y  0x08049031 lab9-2.asm:22
(gdb) 
```

Рис. 2.9: Установка точки останова

С помощью команды `stepi` (или `si`) я пошагово выполняла инструкции, отслеживая изменения регистров.

```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0b0 0xffffd0b0
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 3715 (asm) 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 значений регистров


```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0b0 0xffffd0b0
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 3715 (asm) In: _start L16 PC: 0x8049016
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) 
```

Рис. 2.11: Отслеживание изменений регистров

Я также изменила значение переменной msg1 и регистров, используя команду set.

```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0b0 0xffffd0b0
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 3715 (asm) 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 переменной

```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0b0 0xffffd0b0
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 3715 (asm) 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: Отображение измененного регистра

Используя аналогичные команды, я изменила значение регистра ebx.

```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd0b0 0xffffd0b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

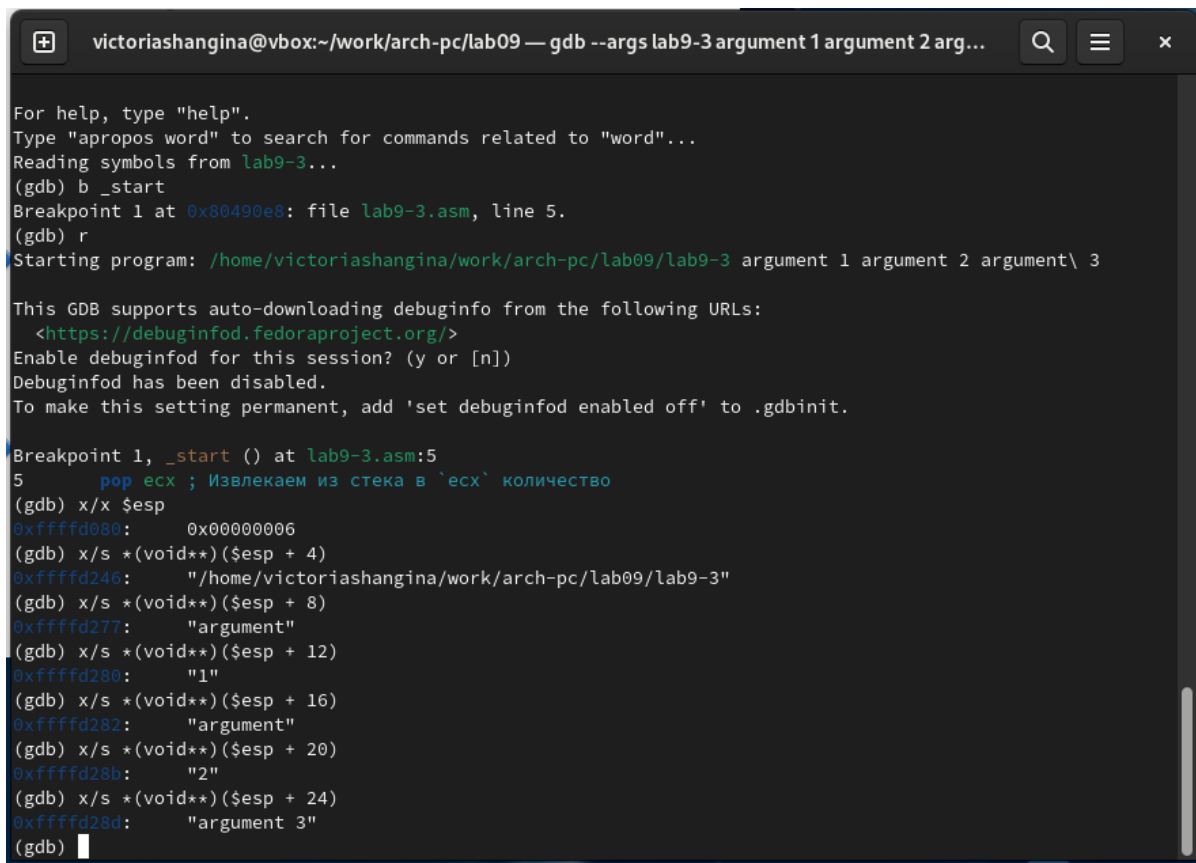
B+ 0x8049000 <_start>    mov     eax,0x4
0x8049005 <_start+5>    mov     ebx,0x1
0x804900a <_start+10>   mov     ecx,0x804a000
0x804900f <_start+15>   mov     edx,0x8
0x8049014 <_start+20>   int     0x80
>0x8049016 <_start+22>  mov     eax,0x4
0x804901b <_start+27>  mov     ebx,0x1
0x8049020 <_start+32>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

native process 3715 (asm) 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: Изменение регистра ebx

2.3 Работа с аргументами командной строки

Для работы с аргументами командной строки я использовала файл lab8-2.asm (из лабораторной работы №8), создав из него исполняемый файл. Затем загрузила программу в GDB с аргументами, используя ключ `-args`.



```
victoriashangina@vbox:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 arg...
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/victoriashangina/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

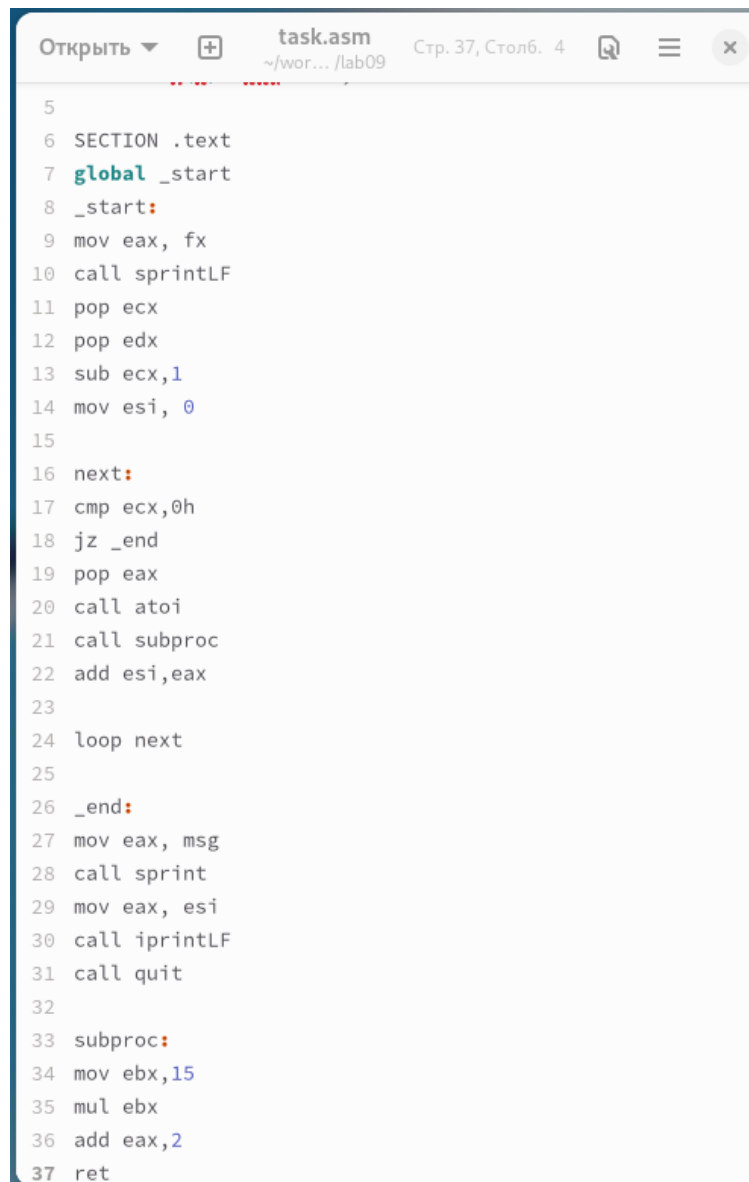
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd080: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd246: "/home/victoriashangina/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd277: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd280: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd282: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd28b: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd28d: "argument 3"
(gdb) 
```

Рис. 2.15: Просмотр аргументов командной строки

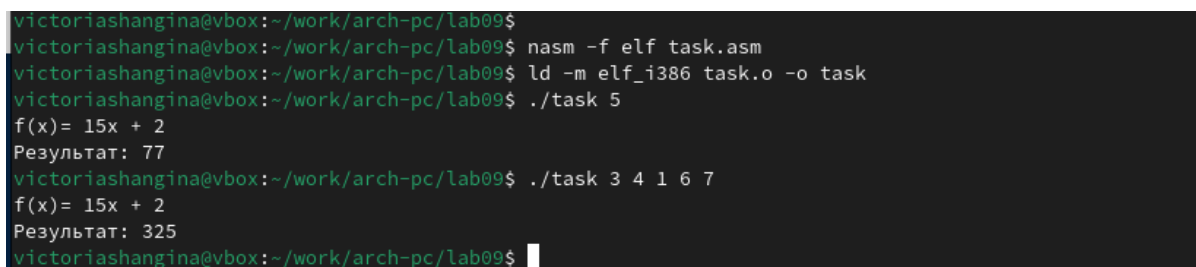
2.4 Задание для самостоятельной работы

В рамках задания я модифицировала программу из лабораторной работы №8, добавив подпрограмму для вычисления функции $f(x)$.



```
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 call subproc
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprint
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 subproc:
34 mov ebx,15
35 mul ebx
36 add eax,2
37 ret
```

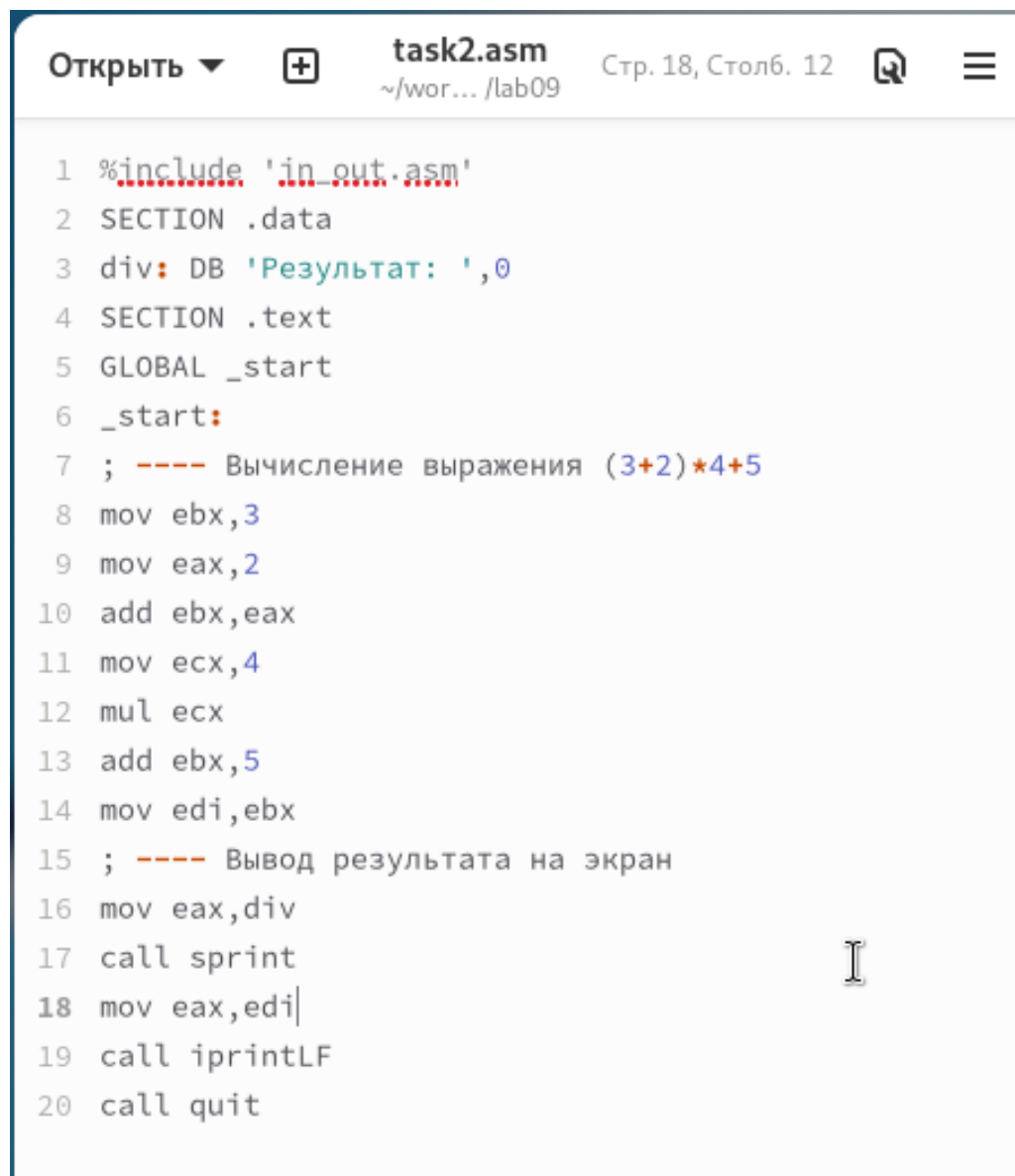
Рис. 2.16: Код программы prog-1.asm



```
victoriashangina@vbox:~/work/arch-pc/lab09$
victoriashangina@vbox:~/work/arch-pc/lab09$ nasm -f elf task.asm
victoriashangina@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 task.o -o task
victoriashangina@vbox:~/work/arch-pc/lab09$ ./task 5
f(x)= 15x + 2
Результат: 77
victoriashangina@vbox:~/work/arch-pc/lab09$ ./task 3 4 1 6 7
f(x)= 15x + 2
Результат: 325
victoriashangina@vbox:~/work/arch-pc/lab09$
```

Рис. 2.17: Результат выполнения программы

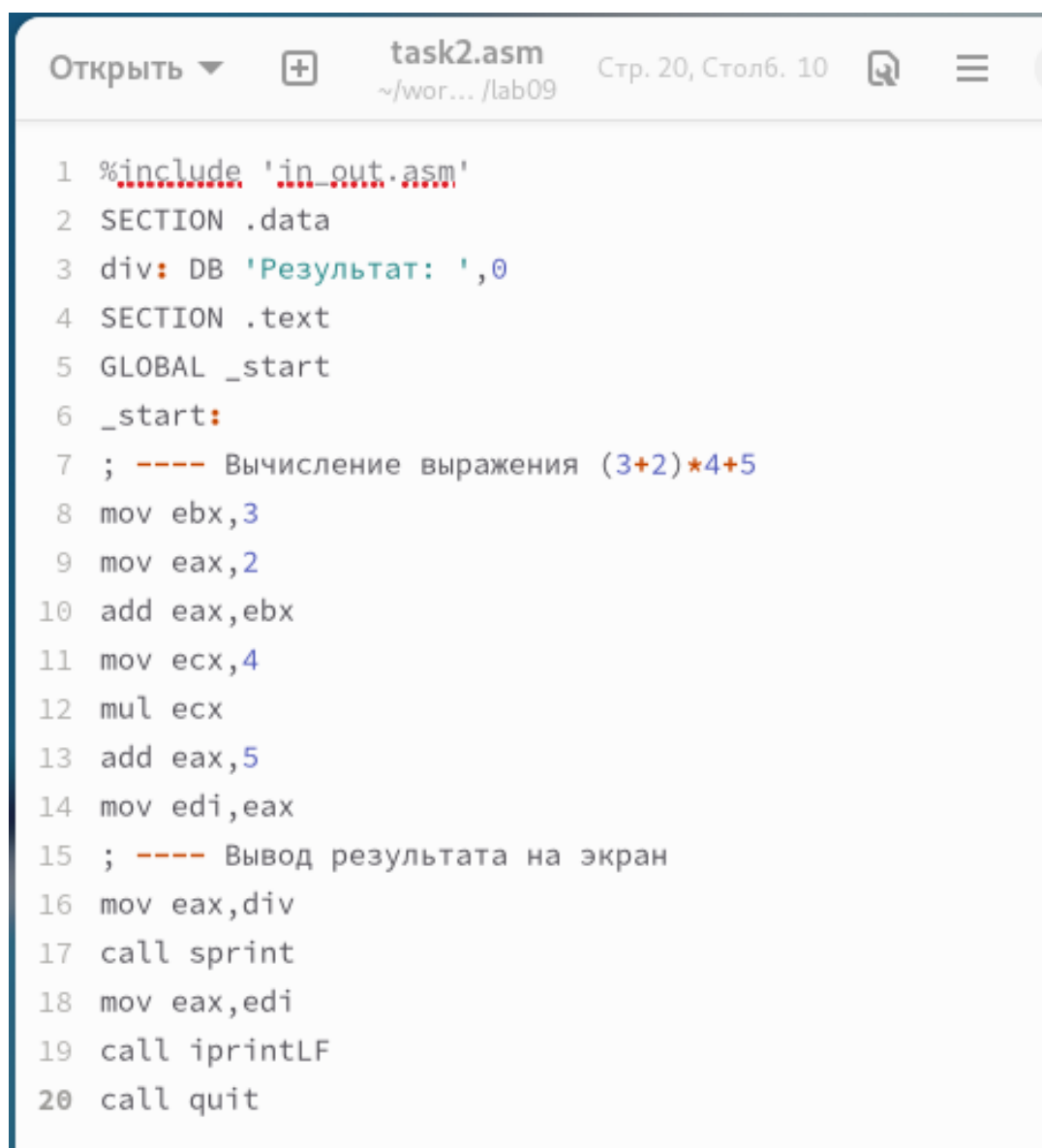
В процессе выполнения программы я обнаружила ошибку: порядок аргументов в инструкции add был перепутан, а регистр ebx вместо eax отправлялся в edi.



```
task2.asm
~/wor.../lab09
Стр. 18, Столб. 12

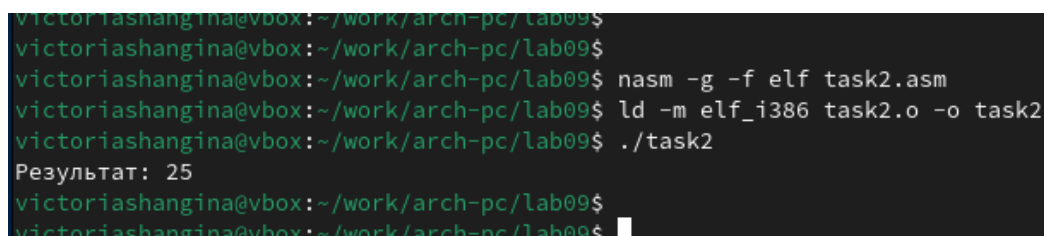
1  %include 'in_out.asm'
2  SECTION .data
3  div: DB 'Результат: ',0
4  SECTION .text
5  GLOBAL _start
6  _start:
7  ; ---- Вычисление выражения (3+2)*4+5
8  mov ebx,3
9  mov eax,2
10 add ebx,eax
11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 ; ---- Вывод результата на экран
16 mov eax,div
17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

Рис. 2.18: Код с ошибками



```
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
```

Рис. 2.20: Исправленный код программы



```
victoriashangina@vbox:~/work/arch-pc/lab09$
victoriashangina@vbox:~/work/arch-pc/lab09$
victoriashangina@vbox:~/work/arch-pc/lab09$ nasm -g -f elf task2.asm
victoriashangina@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 task2.o -o task2
victoriashangina@vbox:~/work/arch-pc/lab09$ ./task2
Результат: 25
victoriashangina@vbox:~/work/arch-pc/lab09$
victoriashangina@vbox:~/work/arch-pc/lab09$
```

Рис. 2.21: Результат проверки

2.5 Выводы

В ходе лабораторной работы я научилась работать с подпрограммами и отладчиком GDB, а также диагностировать и исправлять ошибки в ассемблерных программах.