

Отчёт по лабораторной работе 9

дисциплина: Архитектура компьютера

Алиев Руслан Нияз оглы

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	Задание для самостоятельной работы	20
3	Выводы	27

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

Список таблиц

1 Цель работы

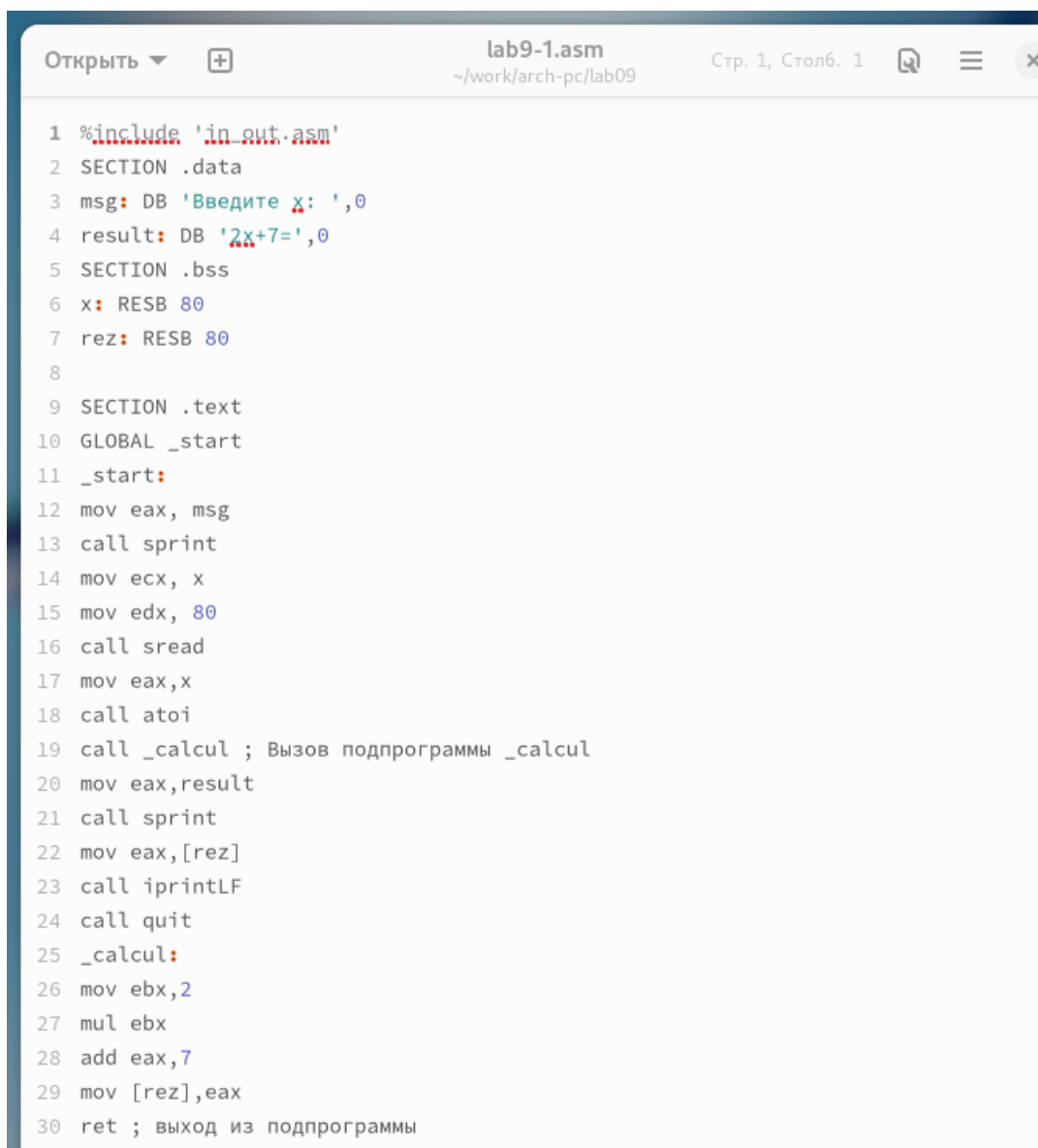
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

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
```

Рис. 2.1: Текст программы lab9-1.asm

```
alievruslan@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 3
2x+7=13
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 5
2x+7=17
alievruslan@VirtualBox:~/work/arch-pc/lab09$ █
```

I

Рис. 2.2: Запуск программы lab9-1.asm

Далее я модифицировал программу, добавив подпрограмму `subcalcul` внутрь 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)



```
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
```

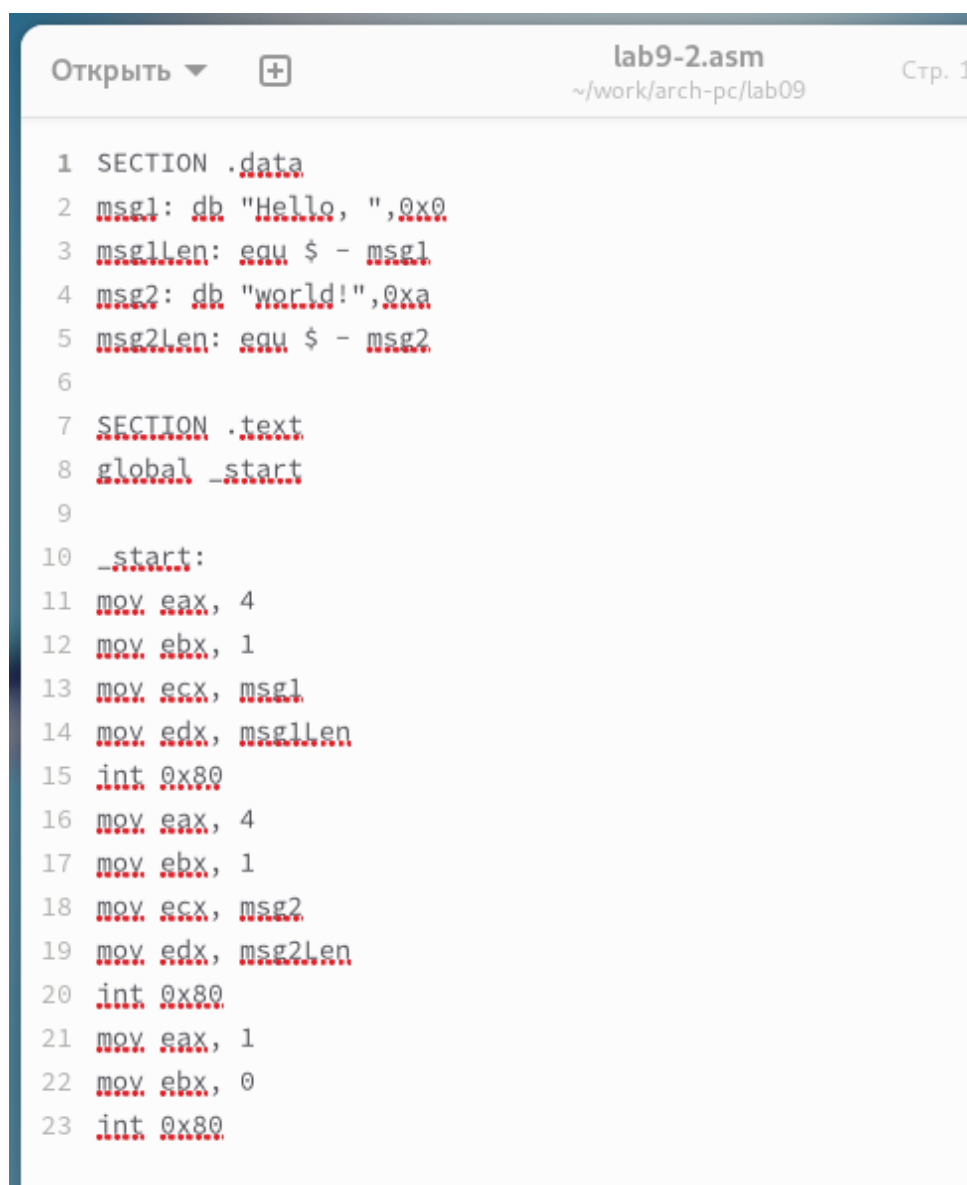
Рис. 2.3: Модифицированная программа lab9-1.asm

```
alievruslan@VirtualBox:~/work/arch-pc/lab09$  
alievruslan@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm  
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o  
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 3  
2(3x-1)+7=23  
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 5  
2(3x-1)+7=35  
alievruslan@VirtualBox:~/work/arch-pc/lab09$
```

Рис. 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
```

Рис. 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)

```

alievruslan@VirtualBox:~/work/arch-pc/lab09$
alievruslan@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
alievruslan@VirtualBox:~/work/arch-pc/lab09$ gdb lab9-2
rGNU 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/alievruslan/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 3755) exited normally]
(gdb) █

```

Рис. 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)

```
alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) r
Starting program: /home/alievruslan/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 3755) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/alievruslan/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 программы

```
alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb 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) 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 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)

The screenshot shows a GDB terminal window with the title bar "alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", lists the values of general-purpose registers: `eax` (0x0), `ecx` (0x0), `edx` (0x0), `ebx` (0x0), `esp` (0xffffd0e0), `ebp` (0x0), `esi` (0x0), `edi` (0x0), `eip` (0x8049000), and `eflags` (0x202). The middle section displays assembly code starting from address 0x8049000, with instructions like `mov eax, 0x4`, `mov ebx, 0x1`, `mov ecx, 0x804a000`, `mov edx, 0x8`, `int 0x80`, and so on. The bottom section shows the GDB command prompt with the following commands and output: `(gdb) layout regs`, `(gdb) b *0x8049031` (setting a breakpoint), `Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.`, `(gdb) i b` (listing breakpoints), and a table of breakpoints. The table has columns for "Num", "Type", "Disp", "Enb", "Address", and "What". It lists two breakpoints: one at 0x08049000 (hit 1 time) and another at 0x08049031.

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0e0 0xffffd0e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]

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

native process 3762 (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: Настройка точки останова

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

```
alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0e0 0xffffd0e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]

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

native process 3762 (asm) In: _start L12 PC: 0x8049005
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb)
```

Рис. 2.10: Отслеживание изменений регистров

The screenshot shows a GDB terminal window with the title bar "alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into several sections:

- Register group: general**: A table showing the values of general-purpose registers.

Register	Value (hex)	Value (dec)
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0e0	0xffffd0e0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>
eflags	0x202	[IF]
- Assembly code**: A list of instructions with their addresses and disassembled forms.

```
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
```
- Process information**: "native process 3762 (asm) In: _start" with "L16" and "PC: 0x8049016".
- Segment registers**:

```
--Type <RET> for more, q to quit, c to continue without paging--
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0       0
gs      0x0       0
```
- GDB prompts**: Several "(gdb) si" commands are shown, indicating step-through execution.

Рис. 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)

The screenshot shows a GDB window titled "alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The top panel displays the "Register group: general" with the following values:

Register	Value	Comment
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0e0	0xffffd0e0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>
eflags	0x202	[IF]

The middle panel shows assembly code with the instruction at address 0x8049016 highlighted:

```
B+ 0x8049000 <_start>    mov     eax,0x4
0x8049005 <_start+5>    mov     ebx,0x1
0x804900a <_start+10>   mov     ecx,0x804a000
0x804900f <_start+15>   mov     edx,0x8
0x8049014 <_start+20>   int     0x80
>0x8049016 <_start+22>  mov     eax,0x4
0x804901b <_start+27>   mov     ebx,0x1
0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7
0x804902a <_start+42>   int     0x80
```

The bottom panel shows the GDB command line with the following commands and output:

```
native process 3762 (asm) In: _start L16 PC: 0x8049016
(gdb) si
(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>:      "Lor!d!\n\034"
(gdb)
```

Рис. 2.12: Изменение значения переменной msg1

The screenshot shows the GDB interface with the title bar 'alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2'. The 'Register group: general' window is open, displaying the following register values:

Register	Value (hex)	Value (dec)
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0e0	0xffffd0e0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>
eflags	0x202	[IF]

Below the register window, the assembly code is displayed, with the instruction at address 0x8049016 highlighted:

```
B+ 0x8049000 <_start>    mov     eax,0x4
0x8049005 <_start+5>    mov     ebx,0x1
0x804900a <_start+10>   mov     ecx,0x804a000
0x804900f <_start+15>   mov     edx,0x8
0x8049014 <_start+20>   int      0x80
>0x8049016 <_start+22>  mov     eax,0x4
0x804901b <_start+27>   mov     ebx,0x1
0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7
0x804902a <_start+42>   int      0x80
```

The bottom panel shows the command prompt with the following text:

```
native process 3762 (asm) In: _start L16 PC: 0x8049016
$2 = 1000
(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: Просмотр регистра после изменений

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. 2.14)

```
alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argume...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/alievruslan/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

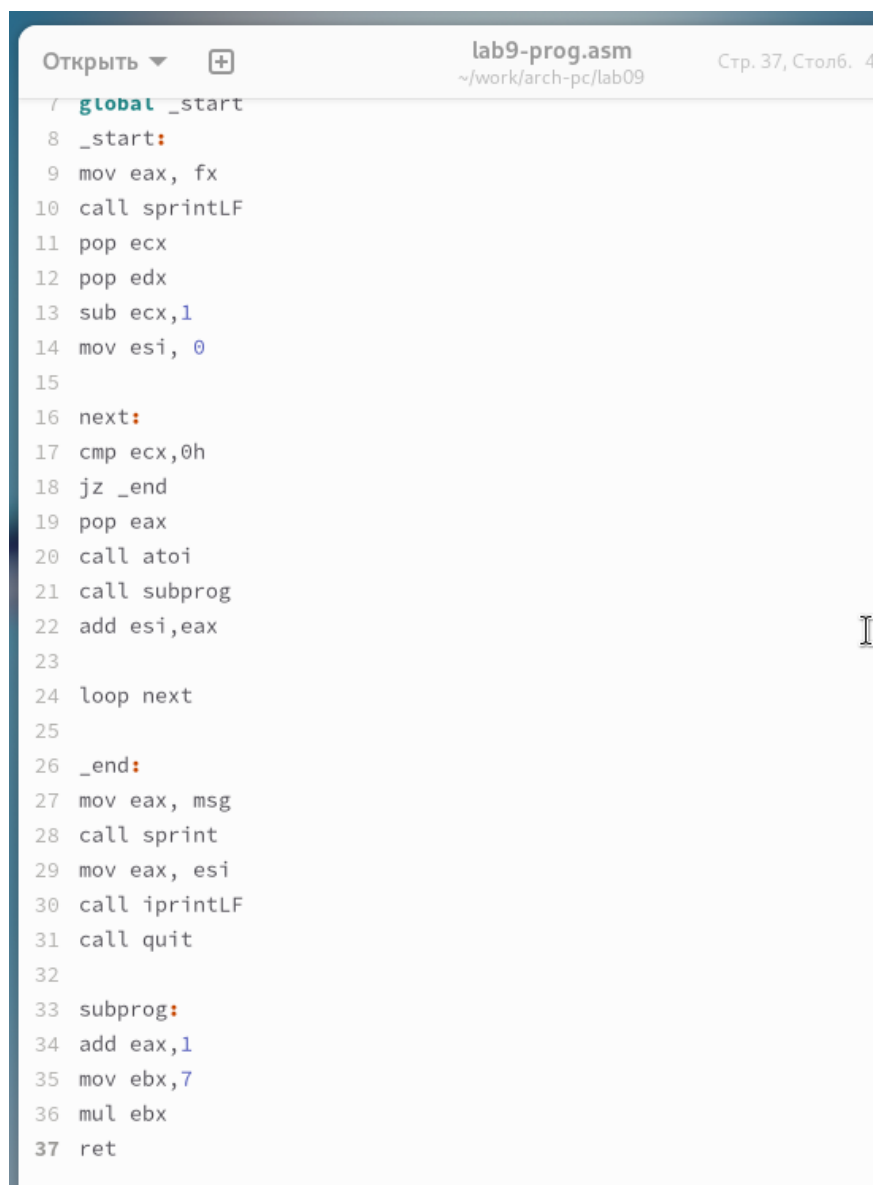
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.


Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd0b0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd277: "/home/alievruslan/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2a3: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2ac: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2ae: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2b7: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2b9: "argument 3"
(gdb) c
Continuing.
argument
1
argument
2
argument 3
[Inferior 1 (process 3799) exited normally]
(gdb) █
```

Рис. 2.14: Анализ стека программы

2.3 Задание для самостоятельной работы

Я модифицировал программу из лабораторной работы №8, добавив вычислительные функции $f(x)$ в виде подпрограммы. (рис. 2.15) (рис. 2.16)



```
Открыть ▾  lab9-prog.asm
~/work/arch-pc/lab09 Стр. 37, Столб. 4

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 subprog
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 subprog:
34 add eax,1
35 mov ebx,7
36 mul ebx
37 ret
```

Рис. 2.15: Код программы lab9-prog.asm

```

alievruslan@VirtualBox:~/work/arch-pc/lab09$
alievruslan@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-prog.asm
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-prog.o -o lab9-prog
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-prog
f(x)= 7(x + 1)
Результат: 0
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-prog 1
f(x)= 7(x + 1)
Результат: 14
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-prog 2
f(x)= 7(x + 1)
Результат: 21
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ./lab9-prog 4 3 1 6 8 7 9
f(x)= 7(x + 1)
Результат: 315
alievruslan@VirtualBox:~/work/arch-pc/lab09$

```

Рис. 2.16: 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. 2.17) (рис. 2.18)

Открыть ▾ 

lab9-prog2.asm
~/work/arch-pc/lab09

Стр. 1, Столб

```
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. 2.17: Код с ошибкой

The screenshot shows a GDB debugger window titled "alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-prog2". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of various CPU registers: `eax` (0x8), `ecx` (0x4), `edx` (0x0), `ebx` (0xa), `esp` (0xffffd0e0), `ebp` (0x0), `esi` (0x0), `edi` (0xa), `eip` (0x8049100), and `eflags` (0x206). The middle section shows assembly code with addresses and instructions: `B+ 0x80490e8 <_start> mov ebx,0x3`, `0x80490ed <_start+5> mov eax,0x2`, `0x80490f2 <_start+10> add ebx,eax`, `0x80490f4 <_start+12> mov ecx,0x4`, `0x80490f9 <_start+17> mul ecx`, `0x80490fb <_start+19> add ebx,0x5`, `0x80490fe <_start+22> mov edi,ebx`, `>0x8049100 <_start+24> mov eax,0x804a000`, `0x8049105 <_start+29> call 0x804900f <sprint>`, and `0x804910a <_start+34> mov eax,edi`. The bottom section shows the status of the native process (3922) and the current instruction pointer (L16, PC: 0x8049100). It also displays a breakpoint at `_start` and a list of commands entered in the GDB prompt.

```
alievruslan@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-prog2

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd0e0 0xffffd0e0
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 <_start+24>
eflags   0x206    [ PF IF ]

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
0x80490f4 <_start+12>   mov     ecx,0x4
0x80490f9 <_start+17>   mul     ecx
0x80490fb <_start+19>   add     ebx,0x5
0x80490fe <_start+22>   mov     edi,ebx
>0x8049100 <_start+24>  mov     eax,0x804a000
0x8049105 <_start+29>  call    0x804900f <sprint>
0x804910a <_start+34>  mov     eax,edi

native process 3922 (asm) In: _start L16 PC: 0x8049100
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-prog2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.18: Процесс отладки программы

После исправления ошибок я проверил работу программы. (рис. 2.19) (рис. 2.20)


```
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|
```

I

Рис. 2.19: Исправленный код программы

```

alievruslan@VirtualBox:~/work/arch-pc/lab09$
alievruslan@VirtualBox:~/work/arch-pc/lab09$ nasm -g -f elf lab9-prog2.asm
alievruslan@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-prog2.o -o lab9-prog2
alievruslan@VirtualBox:~/work/arch-pc/lab09$ gdb lab9-prog2
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-prog2...
(gdb) r
Starting program: /home/alievruslan/work/arch-pc/lab09/lab9-prog2

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.
Результат: 25
[Inferior 1 (process 3971) exited normally]
(gdb) █

```

Рис. 2.20: 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.