

Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Сидельников Андрей Владимирович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работа	17
4	Выводы	22

Список таблиц

Список иллюстраций

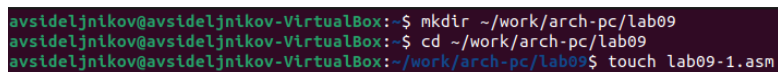
2.1	Создание файла	6
2.2	Листинг9.1	7
2.3	Запуск программы	7
2.4	Изменённый текст программы	8
2.5	Запуск программы	9
2.6	Листинг 9.2 в созданном файле	9
2.7	Запуск программы lab09-2.asm в отладчике	10
figno	Просмотр с помощью команды disassemble _start	10
2.8	Дизассемблированный код в режиме intel	11
2.9	Точка установки	12
2.10	Изменение регистров	12
2.11	Изменение регистров	13
2.12	Изменение значение переменной	13
2.13	Вывод значения регистра	14
2.14	Вывод значения регистра	15
2.15	Вывод значений регистра	16
3.1	Программа в файле lab09-4.asm	18
3.2	Запуск программы lab09-4.asm	18
3.3	Код с ошибкой	19
3.4	Отладка	20
3.5	Изменённая программа	20
3.6	Проверка программы	21

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

Создал каталог для выполнения лабораторной работы № 9 и перешел в него. Затем создал файл lab9-1.asm. (рис. 2.1).

A screenshot of a terminal window with a dark background and light-colored text. It shows three lines of commands and their outputs. The first line creates a directory, the second changes to that directory, and the third creates a new file.

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

Рис. 2.1: Создание файла

Вписываю листинг9.1 в созданный файл (рис. 2.2).

```

lab09-1.asm
1 %include 'in_out.asm' SECTION .data
2 msg: DB 'Введите x: ',0
3 result: DB '2x+7=',0
4 SECTION .bss
5 x: RESB 80
6 res: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 ;-----
11 ; Основная программа
12 ;-----
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
17 call sreadmov eax,x
18 call atoi
19 call _calcul ; Вызов подпрограммы _calcul
20 mov eax,result
21 call sprint
22 mov eax,[res]
23 call iprintLF
24 call quit
25 ;-----
26 ; Подпрограмма вычисления
27 ; выражения "2x+7"
28 _calcul:
29 mov ebx,2
30 mul ebx
31 add eax,7mov [res],eax
32 ret ; выход из подпрограммы

```

Рис. 2.2: Листинг9.1

Запускаю программу из листинга9.1 (рис. 2.3).

```

lab09-1.asm:52: error: expression syntax error
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$ ./lab09-1
Введите x: 8
2x+7=23
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$

```

Рис. 2.3: Запуск программы

Изменил текст программы (рис. 2.4).

```

lab09-1.asm
1 %include 'in_out.asm'
2 SECTION .data
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

```

Рис. 2.4: Изменённый текст программы

Запускаю изменённую программу (рис. 2.5).


```

avsidelnikov@avsidelnikov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
avsidelnikov@avsidelnikov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
avsidelnikov@avsidelnikov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 9
2(3x-1)+7=59

```

Рис. 2.5: Запуск программы

Создал файл lab09-2.asm с текстом программы из Листинга 9.2 (рис. 2.6).

```

lab09-2.asm
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. 2.6: Листинг 9.2 в созданном файле

Получил исполняемый файл и добавил отладочную информацию с помощью ключа '-g' для работы с GDB. Загрузил исполняемый файл в отладчик GDB и проверил работу программы, запустив ее с помощью команды 'run' (рис. 2.7).

```

avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 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-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/avsideljnikov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 3716) exited normally]
(gdb) break _start
Breakpoint 1 at 0x08049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/avsideljnikov/work/arch-pc/lab09/lab09-2

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

```

Рис. 2.7: Запуск программы lab09-2.asm в отладчике

Смотрю дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 2.8).

```

(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.

```

Просмотр с помощью команды `disassemble _start`

```

(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.

```

Рис. 2.8: Дизассемблированный код в режиме intel

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel:

В Порядке операндов:

В АТТ в начале идёт исходный операнд, затем целевой операнд. Например:
`movl %eax, %ebx` (переместить данные из `eax` в `ebx`).

В Intel используется целевой операнд, затем исходный операнд. Например:
`mov ebx, eax`.

Для проверки точки останова по имени метки `'_start'`, использовал команду `'info breakpoints'`. Затем установил еще одну точку останова по адресу инструкции, определив адрес предпоследней инструкции `'mov ebx, 0x0'` (рис. 2.9).

```

[ Register Values Unavailable ]

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
0x8049036 <_start+54> int     0x80
0x8049038 add     BYTE PTR [eax],al

native process 6925 In:  start L9 PC: 0x8049000
(gdb) layout regs
(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  0x8049000 lab09-2.asm:9
      breakpoint already hit 1 time
2      breakpoint       keep y  0x8049031 lab09-2.asm:20
(gdb)

```

Рис. 2.9: Точка установки

Выполнил 5 инструкций с помощью команды stepi и проследил за изменением значений регистров (рис. 2.10) (рис. 2.11).

```

Register group: general
eax     0x4      4
ecx     0x0      0
edx     0x0      0
ebx     0x0      0
esp     0xffffd210 0xffffd210
ebp     0x0      0x0
esi     0x0      0
edi     0x0      0
eip     0x8049005 0x8049005 <_start+5>
eflags  0x202    [ IF ]
cs      0x23     35
ss      0x2b     43

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
0x8049036 <_start+54> int     0x80
0x8049038 add     BYTE PTR [eax],al

native process 6925 In:  start L10 PC: 0x8049005
(gdb) layout regs
(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  0x8049000 lab09-2.asm:9
      breakpoint already hit 1 time
2      breakpoint       keep y  0x8049031 lab09-2.asm:20
(gdb) si
(gdb)

```

Рис. 2.10: Изменение регистров

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd210 0xffffd210
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 < start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
0x8049036 < start+54>   int     0x80
0x8049038             add     BYTE PTR [eax],al

native process 6925 In: start L14 PC: 0x8049016
(gdb) layout regs
(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  0x8049000 lab09-2.asm:9
       breakpoint already hit 1 time
2      breakpoint       keep y  0x8049031 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si

```

Рис. 2.11: Изменение регистров

Просмотрел значение переменной msg1 по имени и получил нужные данные. Просмотрел значение переменной msg1 по имени и получил нужные данные. Для изменения значения регистра или ячейки памяти использовал команду set, указав имя регистра или адрес в качестве аргумента. Изменил первый символ переменной msg1 (рис. 2.12).

```

native process 6925 In: start L14 PC: 0x8049016
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) x/lsb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/lsb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/lsb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/lsb 0x804a008
0x804a008 <msg2>: "Lor!d!\n\034"
(gdb)

```

Рис. 2.12: Изменение значение переменной

Для изменения значения регистра или ячейки памяти использовал команду set, указав имя регистра или адрес в качестве аргумента. Изменил первый символ

переменной msg1 (рис. 2.13)



```
native process 6925 In: start L14 PC: 0x8049016
0x804a000 <msg2>; "Lor1d!\n\034"
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) p/t $eax
$2 = 1000
(gdb) p/x $ecx
$3 = 0x804a000
(gdb) p/s $edx
$4 = 8
(gdb) p/t $edx
$5 = 1000
(gdb) p/x $edx
$6 = 0x8
(gdb) |
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx на нужное значение (рис. 2.14)

```
$1 = 50
(gdb) p/t $eax
$2 = 1000
(gdb) p/x $ecx
$3 = 0x804a000
(gdb) p/s $edx
$4 = 8
(gdb) p/t $edx
$5 = 1000
(gdb) p/x $edx
$6 = 0x8
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb) █
```

Рис. 2.14: Вывод значения регистра

Копирую файл созданный при выполнении лабораторной работы №8 и создаю исполняемый файл

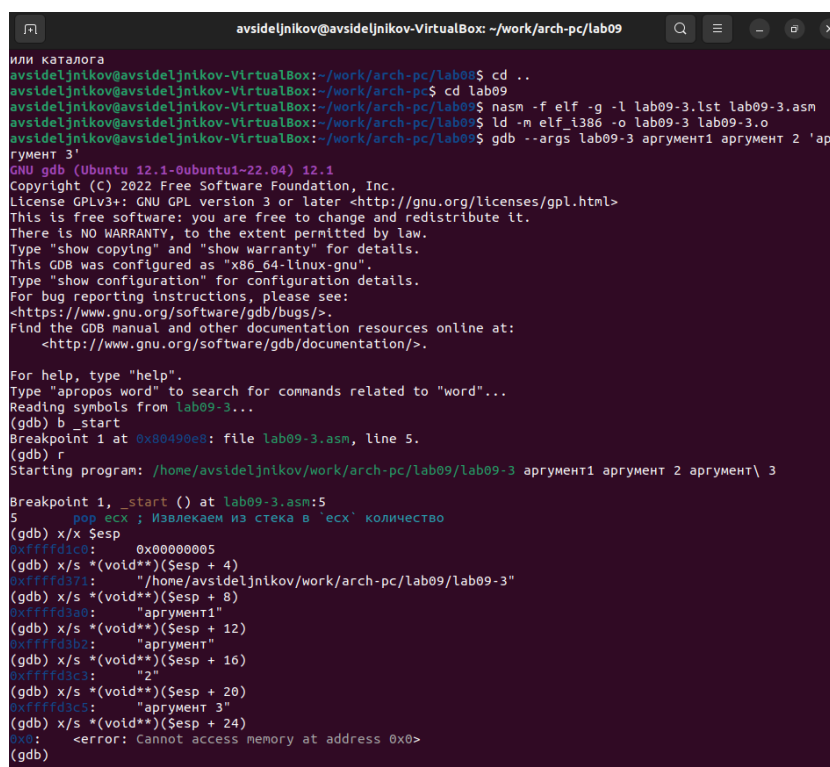
Для загрузки программы с аргументами в gdb использовал ключ `-args` и загрузил исполняемый файл в отладчик с указанными аргументами.

Установил точку останова перед первой инструкцией программы и запустил ее

Адрес вершины стека, содержащий количество аргументов командной строки

(включая имя программы), хранится в регистре `esp`. По этому адресу находится число, указывающее количество аргументов. В данном случае видно, что количество аргументов равно 5, включая имя программы `lab9-3` и сами аргументы: `аргумент1`, `аргумент2` и `'аргумент 3'`.

Просмотрел остальные позиции стека. По адресу `[esp+4]` находится адрес в памяти, где располагается имя программы. По адресу `[esp+8]` хранится адрес первого аргумента, по адресу `[esp+12]` - второго и так далее (рис. 2.15).



```

или каталога
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$ cd ..
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc$ cd lab09
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
avsideljnikov@avsideljnikov-VirtualBox: ~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 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-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) r
Starting program: /home/avsideljnikov/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffd1c0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd371: "/home/avsideljnikov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3a0: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd3b2: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd3c3: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd3c5: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 2.15: Вывод значений регистра

Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего (`[esp+4]`, `[esp+8]`, `[esp+12]`)

3 Самостоятельная работа

Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму (рис. 3.1) (рис. 3.2).

```
*lab09-4.asm
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 7 + 2x',0
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 funk
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 funk:
34 mov ebx,2
35 mul ebx
36 add eax,7
37 ret
```

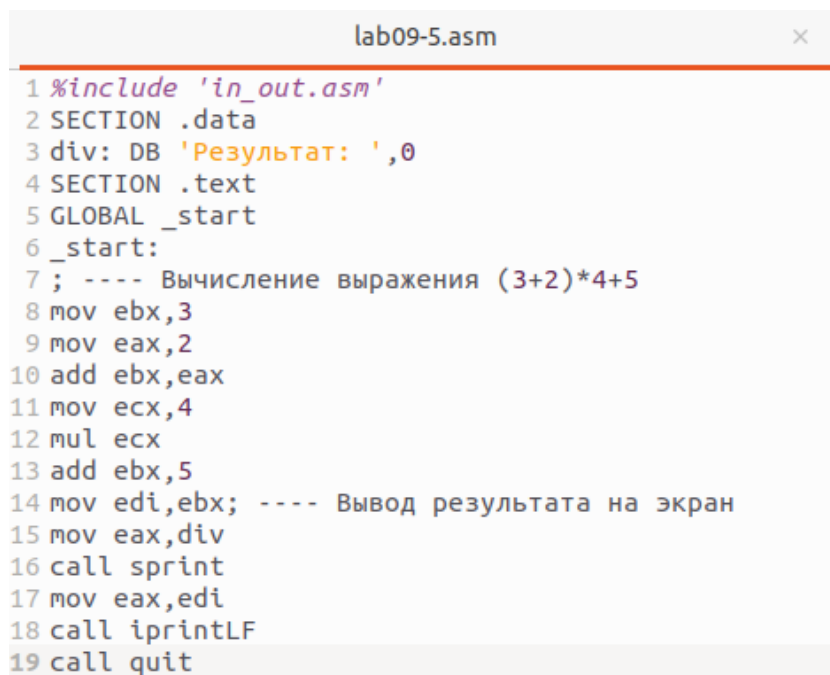
Рис. 3.1: Программа в файле lab09-4.asm

```
avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-4 1
f(x)= 7 + 2x
Результат: 9
avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-4 1 2 6 8
f(x)= 7 + 2x
Результат: 62
avsideljnikov@avsideljnikov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 3.2: Запуск программы lab09-4.asm

В листинге приведена программа вычисления выражения $(3+2)*4+5$. При запуске данная программа дает неверный результат. Проверил это, анализируя изменения значений регистров с помощью отладчика GDB.

Определил ошибку - перепутан порядок аргументов у инструкции add. Также обнаружил, что по окончании работы в edi отправляется ebx вместо eax (рис. 3.3).



```
lab09-5.asm
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; ---- Вывод результата на экран
15 mov eax,div
16 call sprint
17 mov eax,edi
18 call iprintLF
19 call quit
```

Рис. 3.3: Код с ошибкой

Перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax (рис. 3.4).

```

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd200 0xffffd200
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 < start+24>
eflags   0x206    [ PF IF ]
cs       0x23     35
ss       0x2b     43

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 <sprintf>
0x804910a < start+34>  mov     eax,edi
0x804910c < start+36>  call    0x8049086 <printf>
0x8049111 < start+41>  call    0x80490db <quit>
0x8049116             add     BYTE PTR [eax],al
0x8049118             add     BYTE PTR [eax],al

native process 8575 In: start L15 PC: 0x8049100
(gdb) layout regs
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si

```

Рис. 3.4: Отладка

Исправленный код программы (рис. 3.5) (рис. 3.6).

```

lab09-5.asm

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 sprintf
18 mov eax,edi
19 call fprintf
20 call quit

```

Рис. 3.5: Изменённая программа

```
Результат: 25
avsidel'nikov@avsidel'nikov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
avsidel'nikov@avsidel'nikov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
avsidel'nikov@avsidel'nikov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
avsidel'nikov@avsidel'nikov-VirtualBox:~/work/arch-pc/lab09$ █
```

Рис. 3.6: Проверка программы

4 Выводы

Я освоил работу с подпрограммами и отладчиком.