

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра математики и механики

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №9
дисциплина: Архитектура компьютера

Студент: Терёхин Александр Павлович

Группа: НММбД-03-24

МОСКВА

2024г.

Оглавление

1. Цель работы.....	3
2. Задание.....	4
3. Выполнение лабораторной работы.....	5
3 Самостоятельная работа.....	13
4 Вывод.....	17

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.
- 1.2. Задание для самостоятельной работы

В ходе лабораторной работы необходимо научиться реализовывать подпрограммы в NASM и отлаживать программы с помощью GDB.

3. Выполнение лабораторной работы

Я создал каталог lab9 и внутри создал файл lab9-1.asm

```
~/work/study/2023-2024/Архитектура компьютера/lab9
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера $ mkdir lab9
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера $ cd lab 9
bash: cd: слишком много аргументов
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера $ cd lab9
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ touch
lab9-1.asm
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ls
lab9-1.asm
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $
```

Рис. 1: Создание файла lab9-1.asm

Я ввел в файл текст программы и запустил его.

```
lab9-1.asm
~/work/study/2023-2024/Архитектура
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
0     _start:
1 mov eax, msg
2 call sprint
3 mov ecx, x
4 mov edx, 80
5 call sread
6 mov eax,x
7 call atoi
8 call _calcul
9 mov eax,result
0 call sprint
1 mov eax,[res]
2 call iprintLF
3 call quit
4 _calcul:
5     mov ebx,2
6     mul ebx
7     add eax,7
8     mov [res],eax
9     ret
```

Рис. 2: Текст в файле lab9-1.asm

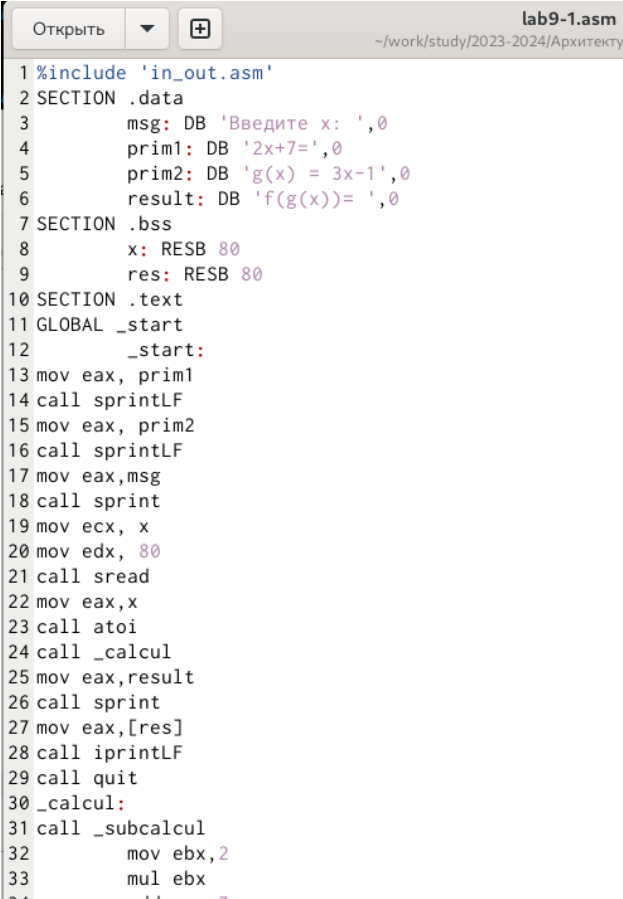
Я создал исполняемый файл и запустил его. Результат соответствовал нужному.



```
gdb lab9-2
r directory
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ nasm
-f elf lab9-1.asm
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ld -m
elf_i386 -o lab9-1 lab9-1.o
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ./lab
9-1
Введите x: 5
2x+7=17
```

Рис. 3: Запуск программы lab9-1

Я изменил текст программы, чтобы она решала выражение $f(g(x))$.



```
lab9-1.asm
~/work/study/2023-2024/Архитекту
1 %include 'in_out.asm'
2 SECTION .data
3     msg: DB 'Введите x: ',0
4     prim1: DB '2x+7=',0
5     prim2: DB 'g(x) = 3x-1',0
6     result: DB 'f(g(x))= ',0
7 SECTION .bss
8     x: RESB 80
9     res: RESB 80
10 SECTION .text
11 GLOBAL _start
12     _start:
13     mov eax, prim1
14     call sprintf
15     mov eax, prim2
16     call sprintf
17     mov eax, msg
18     call sprintf
19     mov ecx, x
20     mov edx, 80
21     call sread
22     mov eax, x
23     call atoi
24     call _calcul
25     mov eax, result
26     call sprintf
27     mov eax, [res]
28     call iprintLF
29     call quit
30 _calcul:
31     call _subcalcul
32     mov ebx, 2
33     mul ebx
34     ; ; ;
```

Рис. 4: Изменение текста

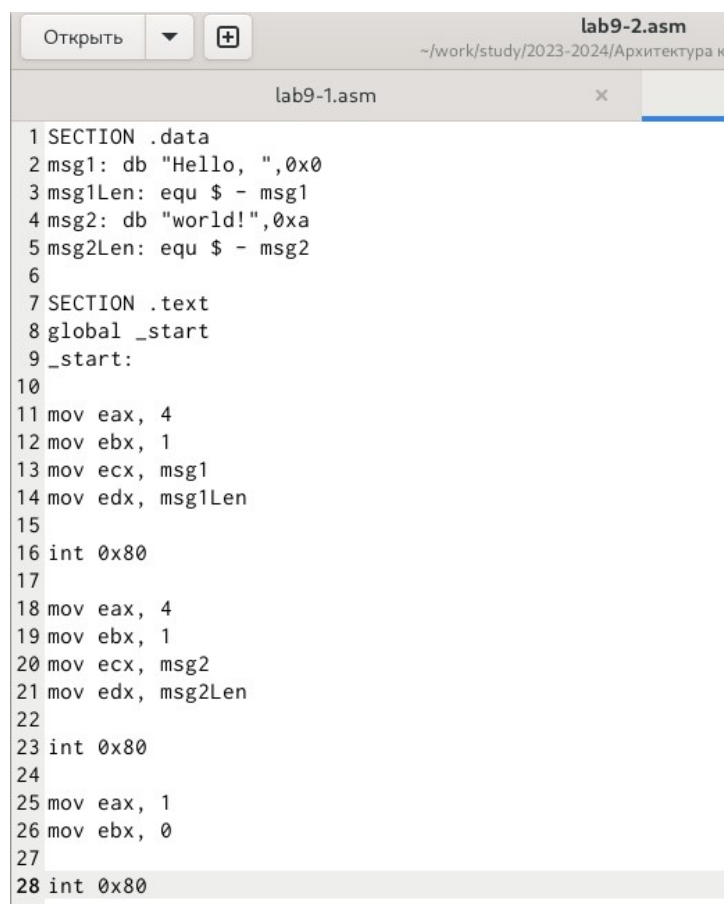
```

apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ nasm
-f elf lab9-1.asm
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ld -m
elf_i386 -o lab9-1 lab9-1.o
apteryokhin@dk3n55 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ./lab
9-1
2x+7=
g(x) = 3x-1
Введите x: 1
f(g(x))= 11

```

Рис. 5: Проверка работы программы

Я создал файл lab9-2.asm и вписал туда программу.



```

lab9-2.asm
~/work/study/2023-2024/Архитектура к
lab9-1.asm
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 _start:
10
11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msg1Len
15
16 int 0x80
17
18 mov eax, 4
19 mov ebx, 1
20 mov ecx, msg2
21 mov edx, msg2Len
22
23 int 0x80
24
25 mov eax, 1
26 mov ebx, 0
27
28 int 0x80

```

Рис. 6: Изменение текста

Запустил программу и проверил ее работу.

```
gdb lab9-2
ab9-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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) 
```

Рис. 7: Отладка второго файла

Я поставил брекпоинт на метку `_start` и запустил программу.

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apteryokhin/work/study/2023-2024/Архитектура компьютера/lab9/lab9-2

Breakpoint 1, _start () at lab9-2.asm:11
11      mov eax, 4
(gdb)
```

Рис. 8: Брекпоинт на метку `_start`

Я просмотрел дисассимплированный код программы начиная с метки.


```

(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. 9: Дисассимплированный код

С помощью команды я переключился на intel'овское отображение синтаксиса. Отличие заключается в командах, в диссимилированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.

```

(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. 10: Intel'овское отображение

Для удобства я включил режим псевдографики.

[Register Values Unavailable]

```
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
0x8049036 <_start+54>   int     0x80
```

: No process In:

L?

b) layout regs

b) █

Я посмотрел наличие меток и добавил еще одну метку на предпоследнюю инструкцию.

```
[ Register Values Unavailable ]

lab9-2.asm
B+> 11 mov eax, 4
    12 mov ebx, 1
    13 mov ecx, msg1
    14 mov edx, msg1Len

native process 5877 In: _start L11 PC: 0x8049000
(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:26
(gdb) 
```

Рис. 11: Наличие меток

С помощью команды `si` я посмотрел регистры и изменил их.

```
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
```

Рис. 12: Измененные регистры

С помощью команды `x` я посмотрел значение переменной `msg1`.

```
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)
```

Рис. 13: Просмотр значения переменной

Следом я посмотрел значение второй переменной `msg2`.

```
(gdb) disassemble _start
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb)
```

Рис. 14: Значение переменной msg2

С помощью команды set я изменил значение переменной msg1.

```
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hhllo, "
(gdb)
```

Рис. 15: Изменение значения переменной

Я изменил переменную msg2.

```
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "Lor d!\n\034"
(gdb)
```

Рис. 18: Изменение msg2

Я вывел значение регистров ecx и eax.

```
(gdb) p/f $msg1
$2 = void
(gdb) p/s $eax
$3 = 4
(gdb) p/t $eax
$4 = 100
(gdb) p/c $ecx
$5 = 0 '\000'
(gdb) p/x $ecx
$6 = 0x0
(gdb)
```

Рис. 19: Значение регистров ecx и eax

Я изменил значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные.

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$7 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$8 = 2
```

Рис. 20: Значение регистров *ebx*

Я завершил работу с файлов вышел.

```
[Inferior 1 (process 3985) exited normally]
```

Рис. 21: Завершение работы с файлов

Я скопировал файл lab9-2.asm и переименовал его. Запустил файл в отладчике и указал аргументы. Затем, поставил метку на `_start`. Я проверил адрес вершины стека и убедился что там хранится 5 элементов.

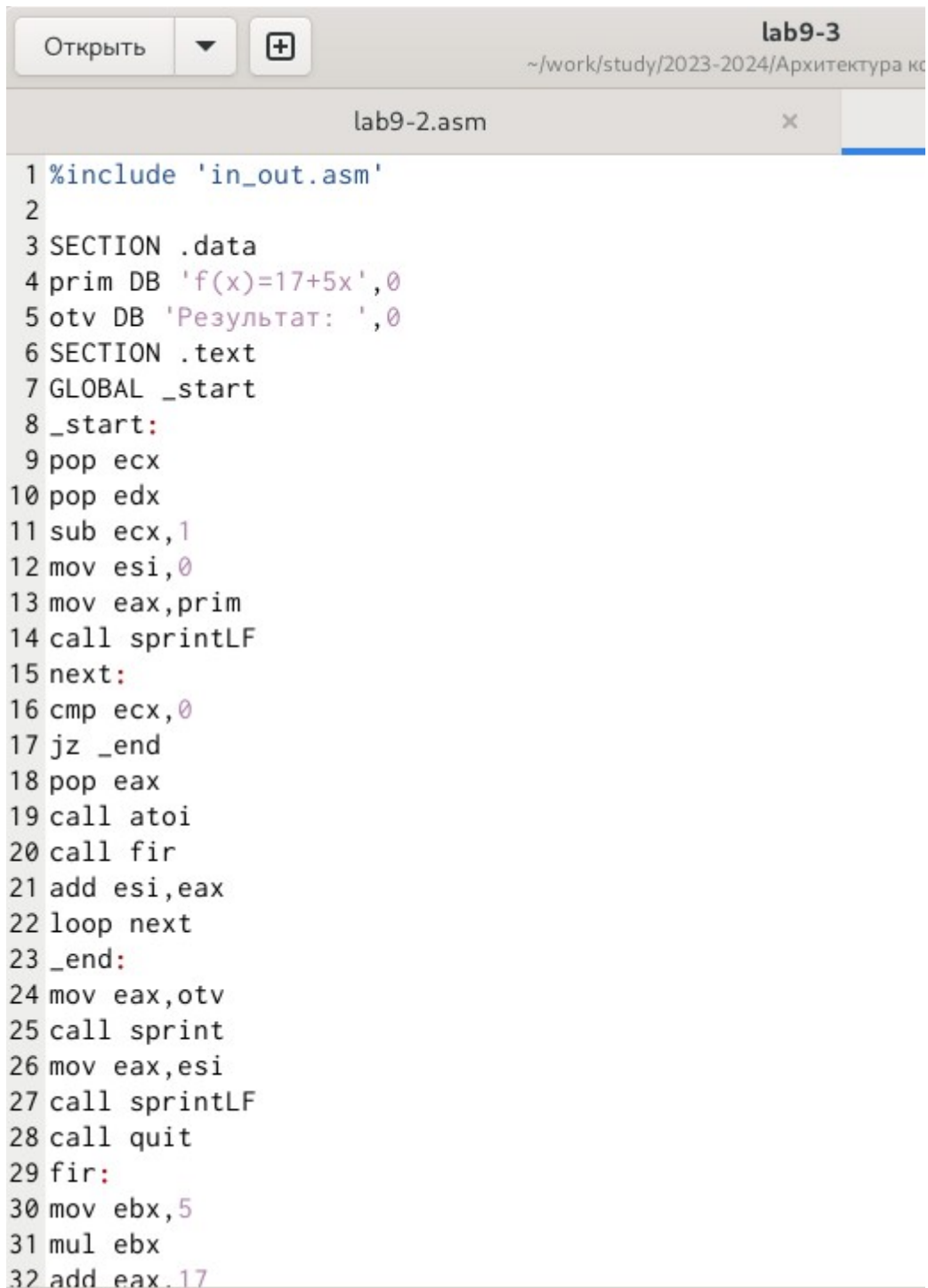
```
(gdb) x/x $esp
0xffffd180: 0x00000005
(gdb)
```

Рис. 22: Адрес вершины стека

Я посмотрел все позиции стека. По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации.

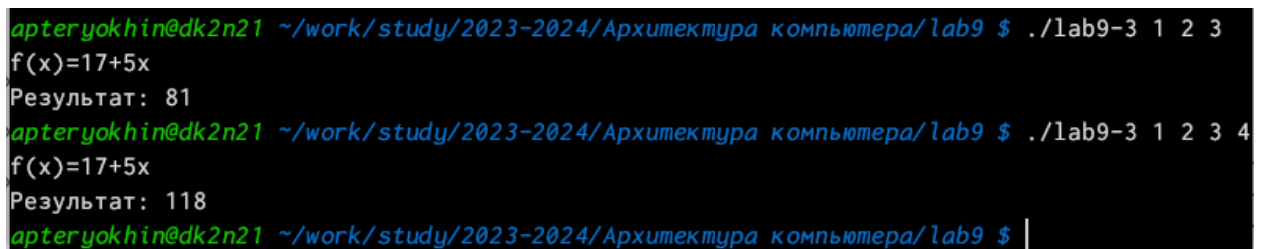
3 Самостоятельная работа.

Я преобразовал программу из лабораторной работы №9 и реализовал вычисления как подпрограмму.



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 prim DB 'f(x)=17+5x',0
5 otv DB 'Результат: ',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 pop ecx
10 pop edx
11 sub ecx,1
12 mov esi,0
13 mov eax,prim
14 call sprintfLF
15 next:
16 cmp ecx,0
17 jz _end
18 pop eax
19 call atoi
20 call fir
21 add esi,eax
22 loop next
23 _end:
24 mov eax,otv
25 call sprintf
26 mov eax,esi
27 call sprintfLF
28 call quit
29 fir:
30 mov ebx,5
31 mul ebx
32 add eax,17
```

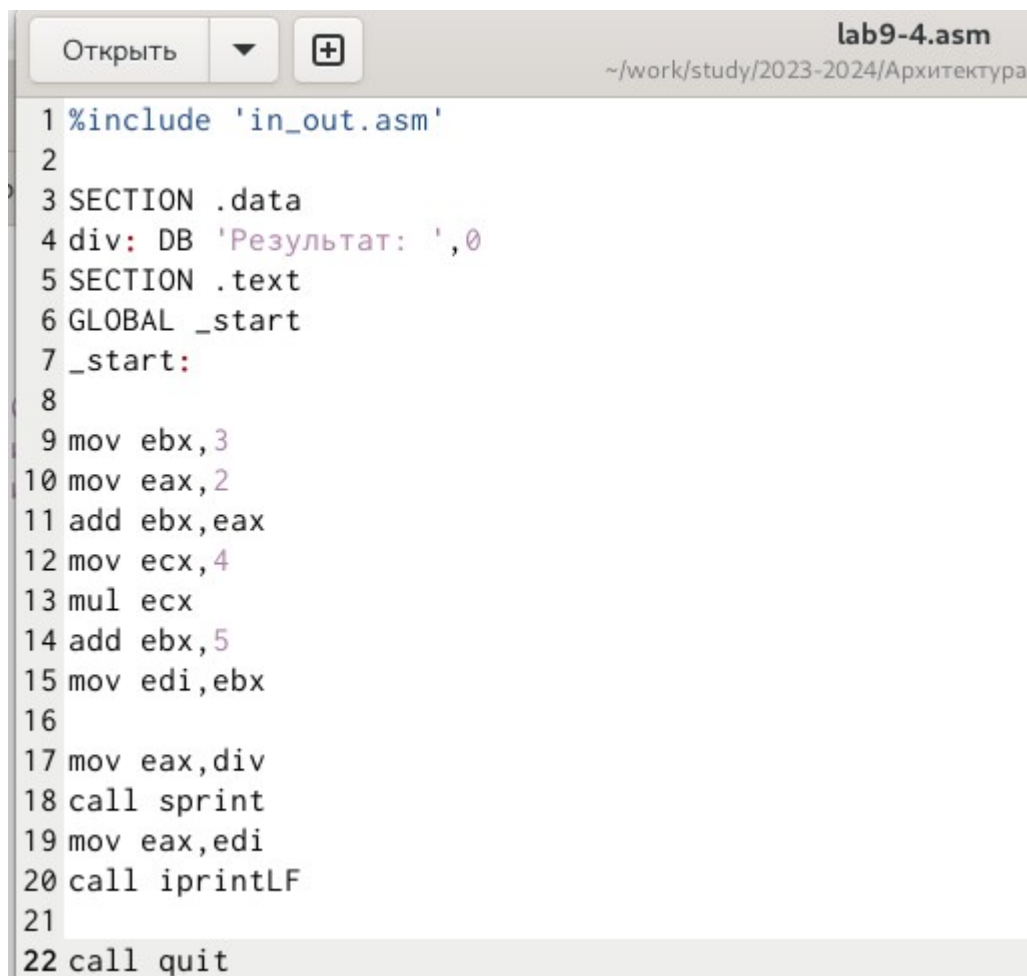
Рис. 23: Текст программы



```
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ./lab9-3 1 2 3
f(x)=17+5x
Результат: 81
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ./lab9-3 1 2 3 4
f(x)=17+5x
Результат: 118
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $
```

Рис. 24: Результат работы программы

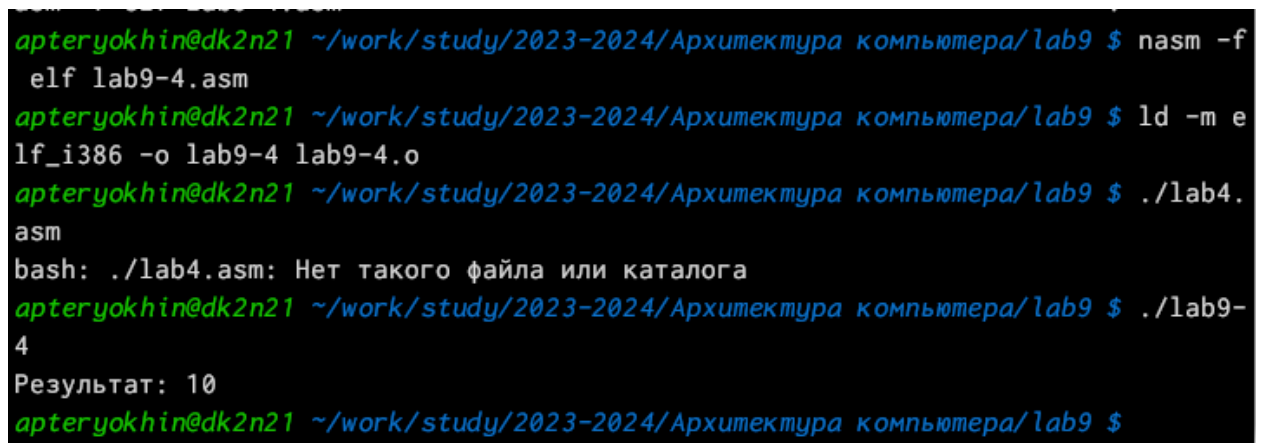
Я переписал программу и попробовал запустить ее чтобы увидеть ошибку. Ошибка была арифметическая, так как вместо 25, программа выводит 10.



```
lab9-4.asm
~/work/study/2023-2024/Архитектура

1 %include 'in_out.asm'
2
3 SECTION .data
4 div: DB 'Результат: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov ebx,3
10 mov eax,2
11 add ebx,eax
12 mov ecx,4
13 mul ecx
14 add ebx,5
15 mov edi,ebx
16
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call iprintLF
21
22 call quit
```

Рис. 25: Текст программы



```
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ nasm -f
elf lab9-4.asm
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ld -m e
lf_i386 -o lab9-4 lab9-4.o
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ./lab4.
asm
bash: ./lab4.asm: Нет такого файла или каталога
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ ./lab9-
4
Результат: 10
apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $
```

Рис. 26: Проверка работы программы

После появления ошибки, я запустил программу в отладчике.

```

apteryokhin@dk2n21 ~/work/study/2023-2024/Архитектура компьютера/lab9 $ gdb lab9-4
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab9-4...
(No debugging symbols found in lab9-4)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apteryokhin/work/study/2023-2024/Архитектура компьютера/lab9/lab9-4

Breakpoint 1, 0x80490e8 in _start ()
(gdb) set disassembly-flavor intel
No symbol table is loaded. Use the "file" command.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x80490e8 <+0>:      mov     ebx,0x3
    0x80490ed <+5>:      mov     eax,0x2
    0x80490f2 <+10>:     add     ebx,eax
    0x80490f4 <+12>:     mov     ecx,0x4
    0x80490f9 <+17>:     mul     ecx
    0x80490fb <+19>:     add     ebx,0x5
    0x80490fe <+22>:     mov     edi,ebx
    0x8049100 <+24>:     mov     eax,0x804a000
    0x8049105 <+29>:     call    0x804900f <sprint>
    0x804910a <+34>:     mov     eax,edi
    0x804910c <+36>:     call    0x8049086 <iprintLF>
    0x8049111 <+41>:     call    0x80490db <quit>
End of assembler dump.
(gdb)

```

Рис. 27: Запуск программы в отладчике

Я открыл регистры и проанализировал их, понял что некоторые регистры стоят не на своих местах и исправил это.


```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x80490db <quit>      mov     ebx,0x0
0x80490e0 <quit+5>    mov     eax,0x1
0x80490e5 <quit+10>   int     0x80
0x80490e7 <quit+12>   ret
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
```

Рис. 28: Анализ регистров

Я изменил регистры и запустил программу, программа вывела ответ 25, то есть все работает правильно.

4 Вывод

Я приобрел навыки написания программ использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями.