

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

Студент: Барханоева Раяна Магометовна
Ст.билет: 1032252468
Группа: НКАбд-01-25

МОСКВА
2025 г

Содержание

1. Цель задачи.....	3
2. Выполнение работы.....	4
3. Самостоятельная работа.....	16
Вывод.....	23

1. Цель задачи

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2. Выполнение работы

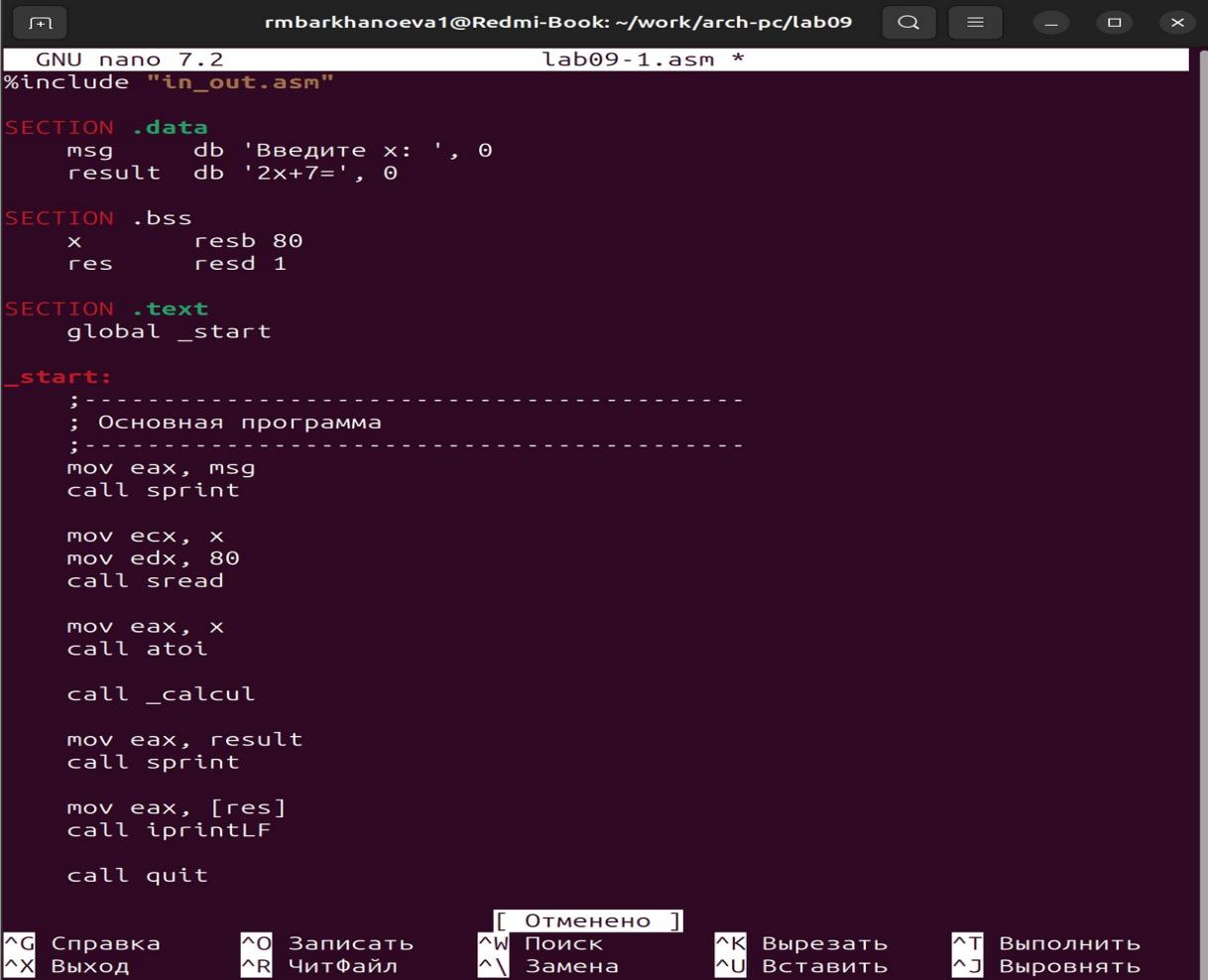
Каталог (рисунок 1).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc$ mkdir lab09
rbarkhanoeva1@Redmi-Book:~/work/arch-pc$ cd lab09/
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ touch lab09-1.asm
```

Рисунок 1.

Создали каталог и файл lab09-1.asm

Программа (рисунок 2).



The screenshot shows the assembly code for a program named `lab09-1.asm`. The code is written in AT&T syntax. It includes sections for data, bss, and text, defines symbols for message and result strings, and implements a main loop to calculate and print the value of $2x+7$.

```
GNU nano 7.2                               lab09-1.asm *
%include "in_out.asm"

SECTION .data
msg      db 'Введите x: ', 0
result   db '2x+7=', 0

SECTION .bss
x        resb 80
res      resd 1

SECTION .text
global _start

_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

call _calcul

mov eax, result
call sprint

mov eax, [res]
call iprintLF

call quit

[ Отменено ]
^G Справка      ^O Записать      ^W Поиск      ^K Вырезать      ^T Выполнить
^X Выход      ^R ЧитФайл      ^\ Замена      ^U Вставить      ^J Выровнять
```

Рисунок 2.

Программа вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы `_calcul`.

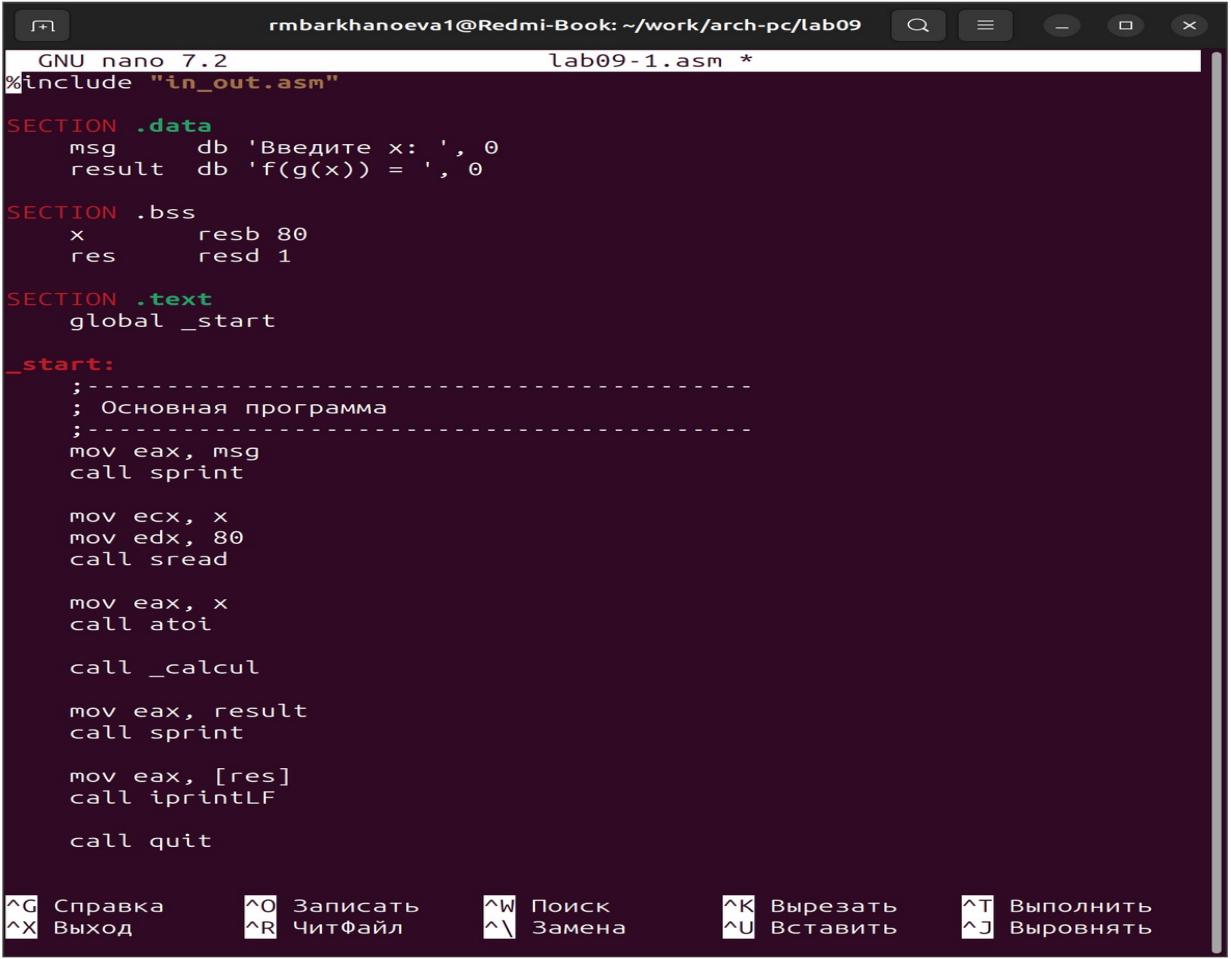
Тест программы (рисунок 3).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1.lab09-1.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ ./lab09-1.
Введите x: 8
2x+7=23
```

Рисунок 3.

Скомпилировали программу и запустили. Ввели число и посчитали формулу.

Программа (рисунок 4).



```
GNU nano 7.2                               lab09-1.asm *
%include "in_out.asm"

SECTION .data
    msg    db 'Введите x: ', 0
    result db 'f(g(x)) = ', 0

SECTION .bss
    x      resb 80
    res    resd 1

SECTION .text
    global _start

_start:
;-----
; Основная программа
;-----
    mov eax, msg
    call sprint

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi

    call _calcul

    mov eax, result
    call sprint

    mov eax, [res]
    call tprintLF

    call quit

^G Справка   ^O Записать   ^W Поиск   ^K Вырезать   ^T Выполнить
^X Выход   ^R ЧитФайл   ^A Замена   ^U Вставить   ^J Выровнять
```

Рисунок 4.

Первые строки программы отвечают за вывод сообщения на экран (call sprint), чтение данных введенных с клавиатуры (call sread) и преобразования введенных данных из символьного вида в численный (call atoi).

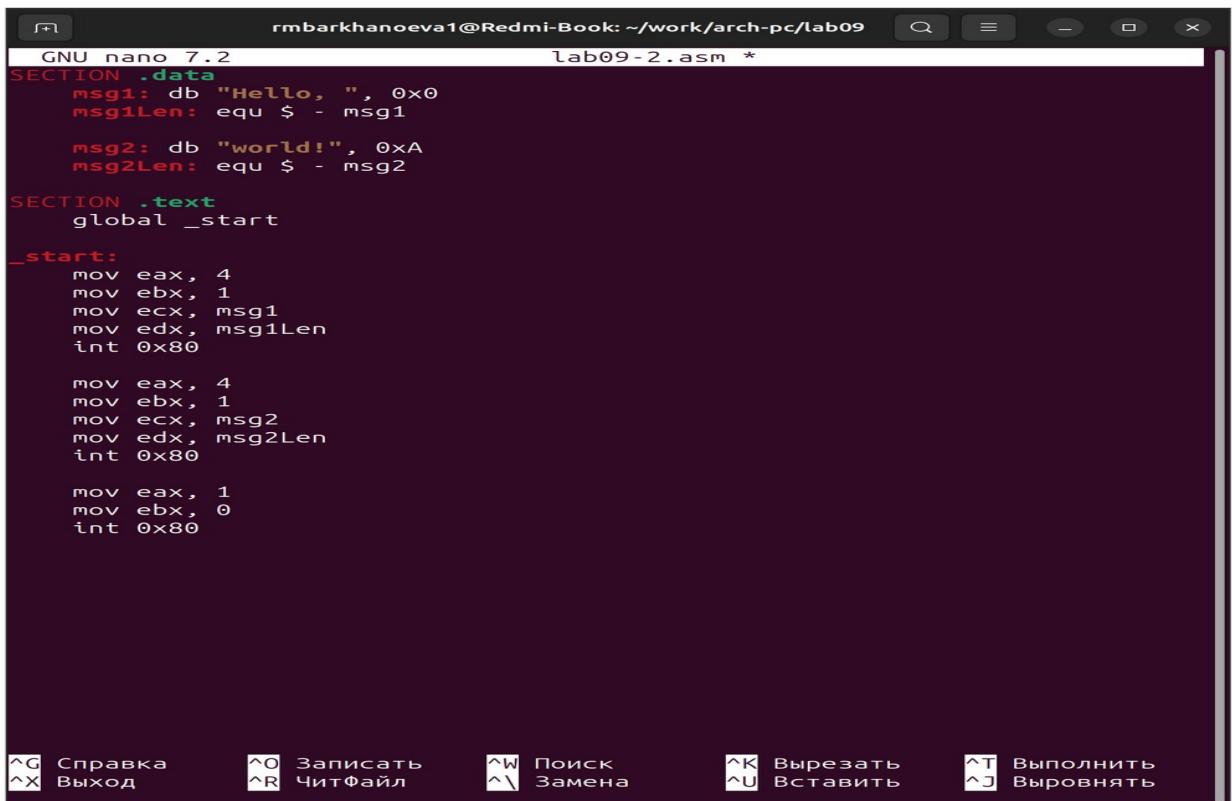
Тест программы (рисунок 5).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1.o
lab09-1.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ ./lab09-1.
Введите x: 8
f(g(x)) = 53
```

Рисунок 5.

Скомпилировали и запустили программу.

Программа (рисунок 6).



```
GNU nano 7.2                               lab09-2.asm *
SECTION .data
    msg1: db "Hello, ", 0x0
    msg1Len: equ $ - msg1

    msg2: db "world!", 0xA
    msg2Len: equ $ - msg2

SECTION .text
    global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, msg1Len
    int 0x80

    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, msg2Len
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Рисунок 6.

Создала файл lab09-2.asm, с текстом программы из Листинга 9.2.
(Программа печати
сообщения Hello world!)

Тест и запуск GDB (рисунок 7).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ nasm -f elf lab09-2.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.l
st lab09-2.asm
ld -m elf_i386 -o lab09-2 lab09-2.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
l>
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)
```

Рисунок 7.

Скомпилировали и запустили программу. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. Загрузили исполняемый файл в отладчик gdb.

GDB (рисунок 8).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
l>
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/rbarkhanoeva1/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit
.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 428361) exited normally]
(gdb)
```

Рисунок 8.

Проверяем работу программы, запустив ее в оболочке GDB с помощью команды run. Видим Hello World.

Брейкпоинт (рисунок 9).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 12.
(gdb) run
Starting program: /home/rbarkhanoeva1/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:12
12      mov eax, 4
(gdb)
```

Рисунок 9.

Для более подробного анализа программы установили брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы и запустили программу

Код программы (рисунок 10).

```
(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)
```

Рисунок 10.

Посмотр дисассимилированного кода программы с помощью команды `disassemble` начиная с метки `_start`.

Код программы (рисунок 11).

```
(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) ■
```

Рисунок 11.

Переключились на отображение команд с Intel'овским синтаксисом, введя команду set disassembly-flavor intel.

Различия отображения ATT и Intel в порядке operandов, регистрах, размерах данных.

Псевдографика (рисунок 12).

```
Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp          0xfffffcf70  0xfffffcf70
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip          0x80490000  <_start>
eflags        0x202      [ IF ]
cs           0x23         35

0x80490ea    add     BYTE PTR [eax],al
0x80490ec    add     BYTE PTR [eax],al
0x80490ee    add     BYTE PTR [eax],al
0x80490f0    add     BYTE PTR [eax],al
0x80490f2    add     BYTE PTR [eax],al
0x80490f4    add     BYTE PTR [eax],al
0x80490f6    add     BYTE PTR [eax],al
0x80490f8    add     BYTE PTR [eax],al
0x80490fa    add     BYTE PTR [eax],al
0x80490fc    add     BYTE PTR [eax],al
0x80490fe    add     BYTE PTR [eax],al
0x8049100    add     BYTE PTR [eax],al

native process 428409 (asm) In: _start          L12   PC: 0x8049000
(gdb) layout regs
(gdb) █
```

Рисунок 12.

Переключился в режим псевдографики для более удобного анализа программы.

Точка (рисунок 13).

```
(gdb) info breakpoints
Num  Type          Disp Enb Address      What
1    bbreakpoint  keep y  0x08049000  lab09-2.asm:12
                                breakpoint already hit 1 time
(gdb) █
```

Рисунок 13.

Установил с помощью команды info breakpoints точку остановы.

Точка (рисунок 14).

```
[gdb] break *0x08049000
(gdb) Note: breakpoint 1 also set at pc 0x08049000.
Breakpoint 3 at 0x08049000: file lab09-2.asm, line 12.
(gdb)
```

Рисунок 14.

Установила еще одну точку остановы по адресу инструкции.

Информация (рисунок 15).

```
(gdb) i b
Num      Type            Disp Enb Address    What
1        breakpoint      keep y  0x08049000  lab09-2.asm:12
          breakpoint already hit 1 time
2        breakpoint      keep y  <PENDING>  0x08049000
3        breakpoint      keep y  0x08049000  lab09-2.asm:12
(gdb) █
```

Рисунок 15.

Информацию о всех установленных точках останова.

Регистр (рисунок 16).

```
Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp 0xfffffcf70 0xfffffcf70
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip 0x80490000 0x8049000 <_start>
eflags        0x202      [ IF ]
cs           0x23         35

0x80490ea    add     BYTE PTR [eax],al
0x80490ec    add     BYTE PTR [eax],al
0x80490ee    add     BYTE PTR [eax],al
0x80490f0    add     BYTE PTR [eax],al
0x80490f2    add     BYTE PTR [eax],al
0x80490f4    add     BYTE PTR [eax],al
0x80490f6    add     BYTE PTR [eax],al
0x80490f8    add     BYTE PTR [eax],al
0x80490fa    add     BYTE PTR [eax],al
0x80490fc    add     BYTE PTR [eax],al
0x80490fe    add     BYTE PTR [eax],al
0x8049100    add     BYTE PTR [eax],al

native process 428409 (asm) In: _start                                L12   PC: 0x8049000
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp 0xfffffcf70 0xfffffcf70
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip 0x80490000 0x8049000 <_start>
eflags        0x202      [ IF ]
cs           0x23         35
ss           0x2b         43
ds           0x2b         43
--Type <RET> for more, q to quit, c to continue without paging--■
```

Рисунок 16.

Просмотр содержимого регистров с помощью команды info registers.

Имя (рисунок 17).

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

Рисунок 17.

Посмотрела значение переменной msg1 по имени.

Адрес (рисунок 18).

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

Рисунок 18.

Посмотрела значение переменной msg2 по адресу.

Изменение (рисунок 19).

```
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) █
```

Рисунок 19.

Изменение первого символа переменной msg1.

Изменение (рисунок 20).

```
(gdb) set {char}0x804a008='h'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "horld!\n\034"
(gdb)
```

Рисунок 20.

С помощью команды set изменила значение регистра.

Изменение (рисунок 21).

```
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$1 = 50  
(gdb) p/s $eax  
$2 = 0  
(gdb)
```

Рисунок 21.

С помощью команды set изменила значение регистра ebx.

p/s \$ebx в GDB выводит значение регистра EBX не как число, а как адрес С-строки, поэтому GDB пытается прочитать память по этому адресу и показать строку до символа \0; если в EBX лежит обычное число, а не адрес строки, вывод будет мусорным или пустым, в отличие от p \$ebx, который всегда показывает числовое значение регистра.

Файлы (рисунок 22).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09/  
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm  
~/work/arch-pc/lab09/lab09-3.asm  
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm  
ld -m elf_i386 -o lab09-3 lab09-3.o  
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ gdb --args lab09-3 5 9 8  
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git  
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-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)
```

Рисунок 22.

С помощью команды cp скопировала файл lab8-2.asm в lab09-3.asm.

Создала исполняемый файл. Загрузила исполняемый файл в отладчик, указав аргументы

Точка (рисунок 23).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 7.
(gdb) run
Starting program: /home/rmbarkhanoeva1/work/arch-pc/lab09/lab09-3 5 9 8

This GDB supports auto-downloading debuginfo from the following URLs:
 <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000

Breakpoint 1, _start () at lab09-3.asm:7
7          pop  ecx      ; Извлекаем из стека в `ecx` количество
(gdb) █
```

Рисунок 23.

Установили точку остановы и запустили ее.

Адрес (рисунок 24).

```
(gdb) x/x $esp
0xfffffcf60: 0x00000004
(gdb) █
```

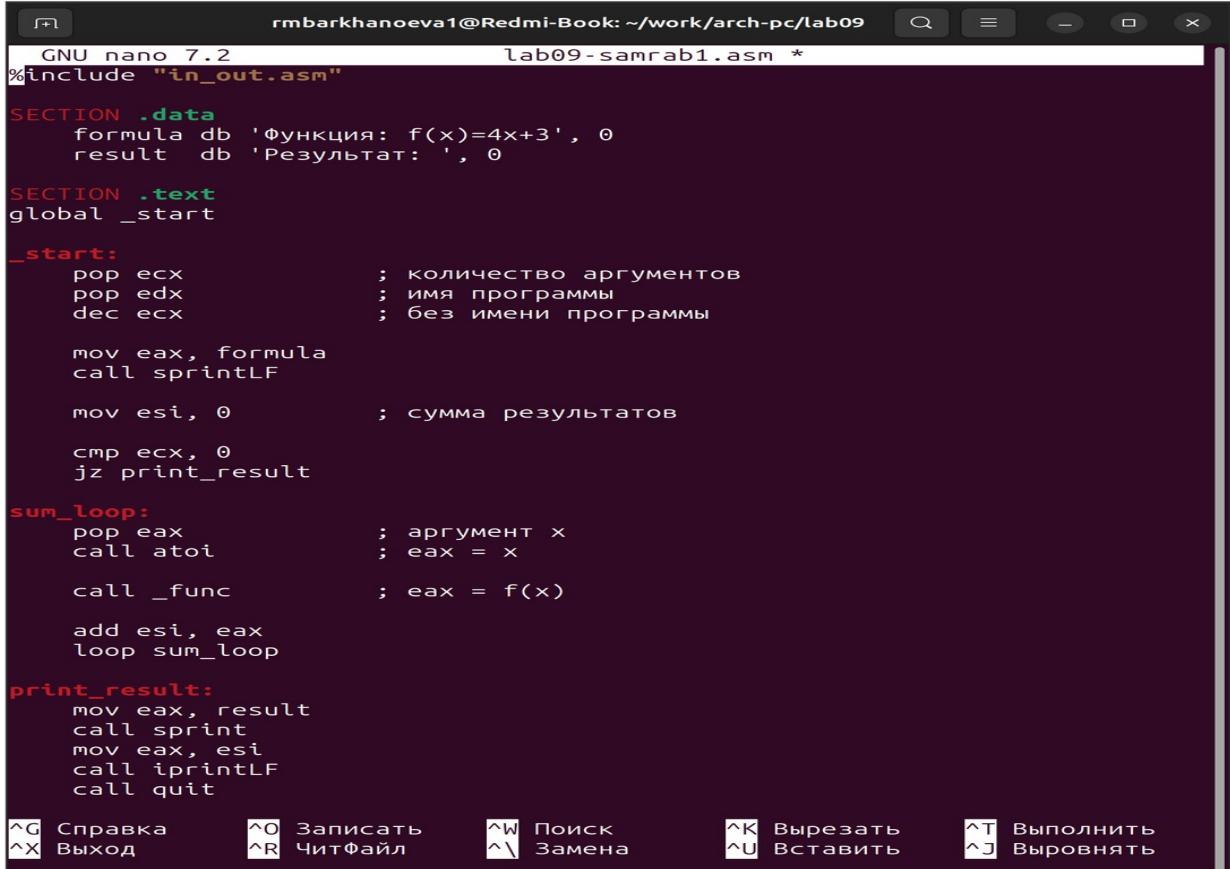
Рисунок 24.

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы).

Шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] и т.д.), потому что в 32-битном режиме x86 размер одного слова (word) и любого адреса/целого регистра составляет 4 байта, стек выровнен по 4 байтам, и каждая операция push или pop помещает или извлекает 4-байтовое значение, поэтому каждый следующий элемент в стеке находится на смещении, кратном 4.

3. Самостоятельная работа

Программа (рисунок 25).



```
GNU nano 7.2          rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab09      lab09-samrab1.asm *
%include "in_out.asm"

SECTION .data
    formula db 'Функция: f(x)=4x+3', 0
    result  db 'Результат: ', 0

SECTION .text
global _start

_start:
    pop ecx           ; количество аргументов
    pop edx           ; имя программы
    dec ecx           ; без имени программы

    mov eax, formula
    call sprintLF

    mov esi, 0         ; сумма результатов

    cmp ecx, 0
    jz print_result

sum_loop:
    pop eax           ; аргумент x
    call atoi          ; eax = x

    call _func          ; eax = f(x)

    add esi, eax
    loop sum_loop

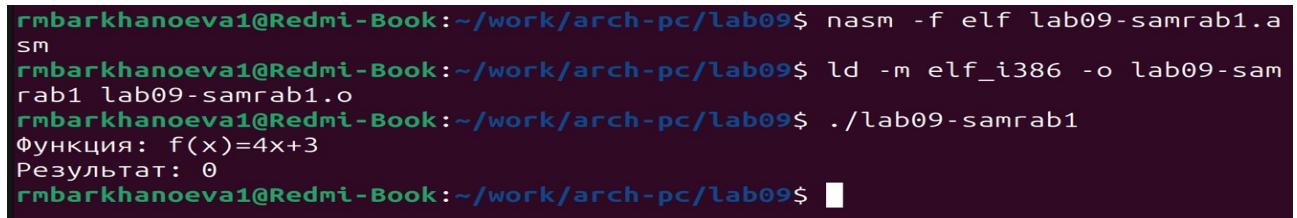
print_result:
    mov eax, result
    call sprint
    mov eax, esi
    call iprintLF
    call quit

^G Справка      ^O Записать      ^W Поиск      ^K Вырезать      ^T Выполнить
^X Выход        ^R ЧитФайл      ^\ Замена      ^U Вставить      ^J Выровнять
```

Рисунок 25.

Программа реализовывает вычисление значения функции $f(x)$ как подпрограмму.

Тест программы (рисунок 26).



```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ nasm -f elf lab09-samrab1.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-samrab1 lab09-samrab1.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ ./lab09-samrab1
Функция: f(x)=4x+3
Результат: 0
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ █
```

Рисунок 26.

Компиляция и тест программы.

Оболочка (рисунок 27).

```
rbarkhaneva1@Redmi-Book:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-sam  
rab2.lst lab09-samrab2.asm  
rbarkhaneva1@Redmi-Book:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-sam  
rab2 lab09-samrab2.o  
rbarkhaneva1@Redmi-Book:~/work/arch-pc/lab09$ gdb lab09-samrab2  
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git  
Copyright (C) 2024 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
l>  
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-samrab2...  
(gdb)
```

Рисунок 27.

Запускаем оболочку gdb

Поинт (рисунок 28).

```
(gdb) break _start  
Breakpoint 1 at 0x80490e8: file lab09-samrab2.asm, line 12.  
(gdb) info breakpoints  
Num      Type            Disp Enb Address     What  
1        breakpoint       keep y    0x080490e8 lab09-samrab2.asm:12  
(gdb) run  
Starting program: /home/rbarkhaneva1/work/arch-pc/lab09/lab09-samrab2  
  
This GDB supports auto-downloading debuginfo from the following URLs:  
<https://debuginfod.ubuntu.com>  
Enable debuginfod for this session? (y or [n]) y  
Debuginfod has been enabled.  
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit  
.br  
Breakpoint 1, _start () at lab09-samrab2.asm:12  
12      mov  ebx,3  
(gdb)
```

Рисунок 28.

Устанавливаем брейкпоинт на `_start` и запускаем программу run

Регистры (рисунок 29).

```
Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x3          3
esp 0xfffffcf90 0xfffffcf90
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip 0x80490ed 0x80490ed <_start+5>
eflags        0x202      [ IF ]
cs           0x23         35

B+ 0x80490e8 <_start>    mov    $0x3,%ebx
>0x80490ed <_start+5>    mov    $0x2,%eax
0x80490f2 <_start+10>   add    %eax,%ebx
0x80490f4 <_start+12>   mov    $0x4,%ecx
0x80490f9 <_start+17>   mul    %ecx
0x80490fb <_start+19>   add    $0x5,%ebx
0x80490fe <_start+22>   mov    %ebx,%edi
0x8049100 <_start+24>   mov    $0x804a000,%eax
0x8049105 <_start+29>   call   0x804900f <sprint>
0x804910a <_start+34>   mov    %edi,%eax
0x804910c <_start+36>   call   0x8049086 <iprintf>
0x8049111 <_start+41>   call   0x80490db <quit>

native process 431667 (asm) In: _start          L13    PC: 0x80490ed
(gdb) layout regs
(gdb) stepi
(gdb)
```

Рисунок 29.

Смотрим значения регистров на старте. Переходим к пошаговому выполнению инструкций `stepi`.

Регистры (рисунок 30).

The screenshot shows a terminal window with the following content:

```
Register group: general
eax          0x2          2
ecx          0x0          0
edx          0x0          0
ebx          0x5          5
esp 0xfffffcf90 0xfffffcf90
ebp 0x0          0x0
esi 0x0          0
edi 0x0          0
eip 0x80490f4 0x80490f4 <_start+12>
eflags 0x206 [ PF IF ]
cs   0x23         35
ss   0x2b         43

B+ 0x80490e8 <_start>    mov    $0x3,%ebx
0x80490ed <_start+5>    mov    $0x2,%eax
0x80490f2 <_start+10>   add    %eax,%ebx
>0x80490f4 <_start+12>  mov    $0x4,%ecx
0x80490f9 <_start+17>   mul    %ecx
0x80490fb <_start+19>   add    $0x5,%ebx
0x80490fe <_start+22>   mov    %ebx,%edi
0x8049100 <_start+24>   mov    $0x804a000,%eax
0x8049105 <_start+29>   call   0x804900f <sprint>
0x804910a <_start+34>   mov    %edi,%eax
0x804910c <_start+36>   call   0x8049086 <iprintfLF>

native process 431667 (asm) In: _start          L15      PC: 0x80490f4
(gdb) info registers eax ebx ecx edx
eax          0x2          2
ebx          0x3          3
ecx          0x0          0
edx          0x0          0
(gdb) (gdb) stepi
Undefined command: "".
Try "help".
(gdb) stepi
(gdb) info registers eax ebx ecx edx
eax          0x2          2
ebx          0x5          5
ecx          0x0          0
edx          0x0          0
(gdb)
```

Рисунок 30.

После каждой инструкции проверяем регистры. Ошибка. После add ebx, eax → EBX = 5, но EAX = 2. Инструкция mul ecx умножает EAX, а ECX = 0 → результат = 0. Поэтому весь результат будет неверный.

Программа (рисунок 31).

```
GNU nano 7.2          lab09-samrab2.asm *
```

```
%include 'in_out.asm'

SECTION .data
div:
    DB 'Результат: ',0

SECTION .text
GLOBAL _start

_start:
; ---- Вычисление выражения (3+2)*4+5
    mov ebx,3
    mov eax,2
    add ebx,eax
    mov eax,ebx      ; перенос суммы в EAX
    mov ecx,4
    mul ecx         ; EAX = (3+2)*4 = 20
    add eax,5       ; EAX = 25
    mov edi,eax      ; сохраняем результат

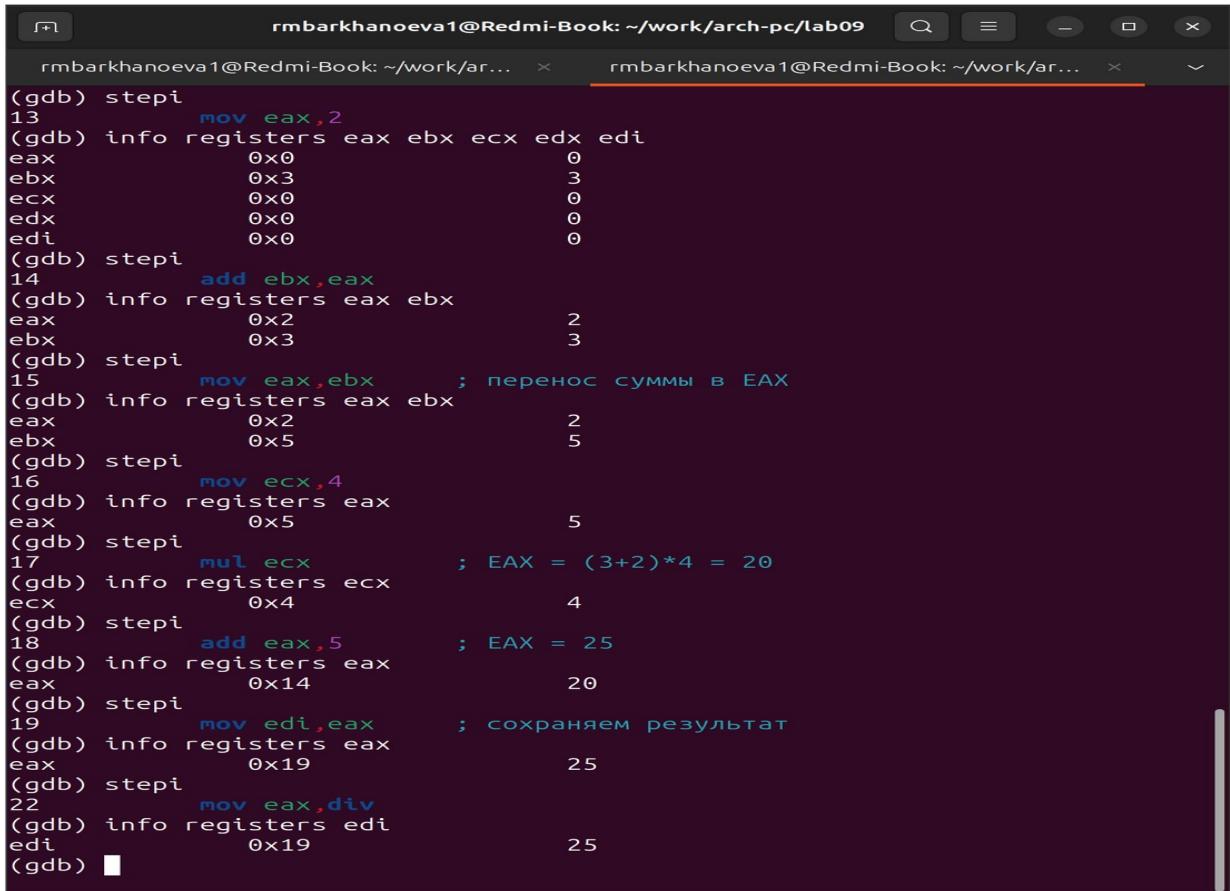
; ---- Вывод результата
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

Справка ^{^G} Выход ^{^X} Записать ^{^O} ЧитФайл ^{^R} Поиск ^{^W} Замена ^{^A} Вырезать ^{^K} Вставить ^{^U} Выполнить ^{^T} Выровнять ^{^J}

Рисунок 31.

В исходном файле добавляем перенос суммы в ЕАХ перед умножением и созраняем результат.

Инструкции (рисунок 32).



The screenshot shows a terminal window with two tabs. The current tab displays assembly code being executed step-by-step in GDB. The code performs a series of operations: it adds EBX and EAX (resulting in EBX = 5), then moves EBX to EAX (resulting in EAX = 5). It then multiplies EAX by 4 (resulting in EAX = 20), adds 5 to EAX (resulting in EAX = 25), and finally divides EAX by 5 (resulting in EDI = 5). The registers are monitored at each step.

```
(gdb) stepi
13      mov eax,2
(gdb) info registers eax ebx ecx edx edi
eax            0x0          0
ebx            0x3          3
ecx            0x0          0
edx            0x0          0
edi            0x0          0
(gdb) stepi
14      add ebx,eax
(gdb) info registers eax ebx
eax            0x2          2
ebx            0x3          3
(gdb) stepi
15      mov eax,ebx      ; перенос суммы в EAX
(gdb) info registers eax ebx
eax            0x2          2
ebx            0x5          5
(gdb) stepi
16      mov ecx,4
(gdb) info registers eax
eax            0x5          5
(gdb) stepi
17      mul ecx        ; EAX = (3+2)*4 = 20
(gdb) info registers ecx
ecx            0x4          4
(gdb) stepi
18      add eax,5      ; EAX = 25
(gdb) info registers eax
eax            0x14         20
(gdb) stepi
19      mov edi,eax    ; сохраняем результат
(gdb) info registers eax
eax            0x19         25
(gdb) stepi
22      mov eax,div
(gdb) info registers edi
edi            0x19         25
(gdb) ■
```

Рисунок 32.

Сохранили код и теперь пошагово выполняем инструкций.

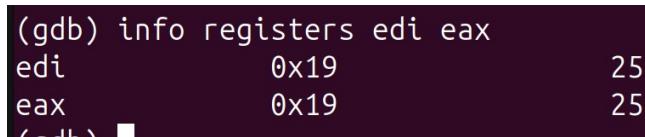
После add ebx,eax → EBX = 5

После mov eax,ebx → EAX = 5

После mul ecx → EAX = 20

После add eax,5 → EAX = 25

Проверка (рисунок 33).



The screenshot shows the terminal displaying the final state of the registers after the execution of the assembly code. The value 25 is present in both the EDI and EAX registers, confirming the correctness of the calculation.

```
(gdb) info registers edi eax
edi            0x19         25
eax            0x19         25
(gdb) ■
```

Рисунок 33.

Проверили.

Завершение (рисунок 34).

```
(gdb) continue  
Continuing.  
Результат: 25  
[Inferior 1 (process 435306) exited normally]
```

Рисунок 34.

Завершение программы и вывод результата.

Вывод (рисунок 35).

```
(gdb) quit  
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$ ./lab09-samrab2  
Результат: 25  
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab09$
```

Рисунок 35.

Вывод

В ходе лабораторной работы была изучена работа с 32-битным ассемблером x86, вычисление арифметического выражения с использованием регистров общего назначения, а также пошаговая отладка программы в GDB; было закреплено понимание порядка выполнения инструкций, влияния каждой команды на значения регистров, принципов работы инструкции `mul`, особенностей хранения данных в регистрах и стеке, выравнивания стека по 4 байта, а также получены практические навыки использования команд `stepi`, `info registers` и `print` для анализа и поиска ошибок, в результате чего программа была корректно отлажена и выдала правильный результат вычислений.