

Отчёт по лабораторной работе №9

Тарутина Кристина Олеговна

Содержание

1	Цель работы	1
2	Выполнение лабораторной работы.....	1
3	Выполнение самостоятельной работы.....	8
4	Выводы	9

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

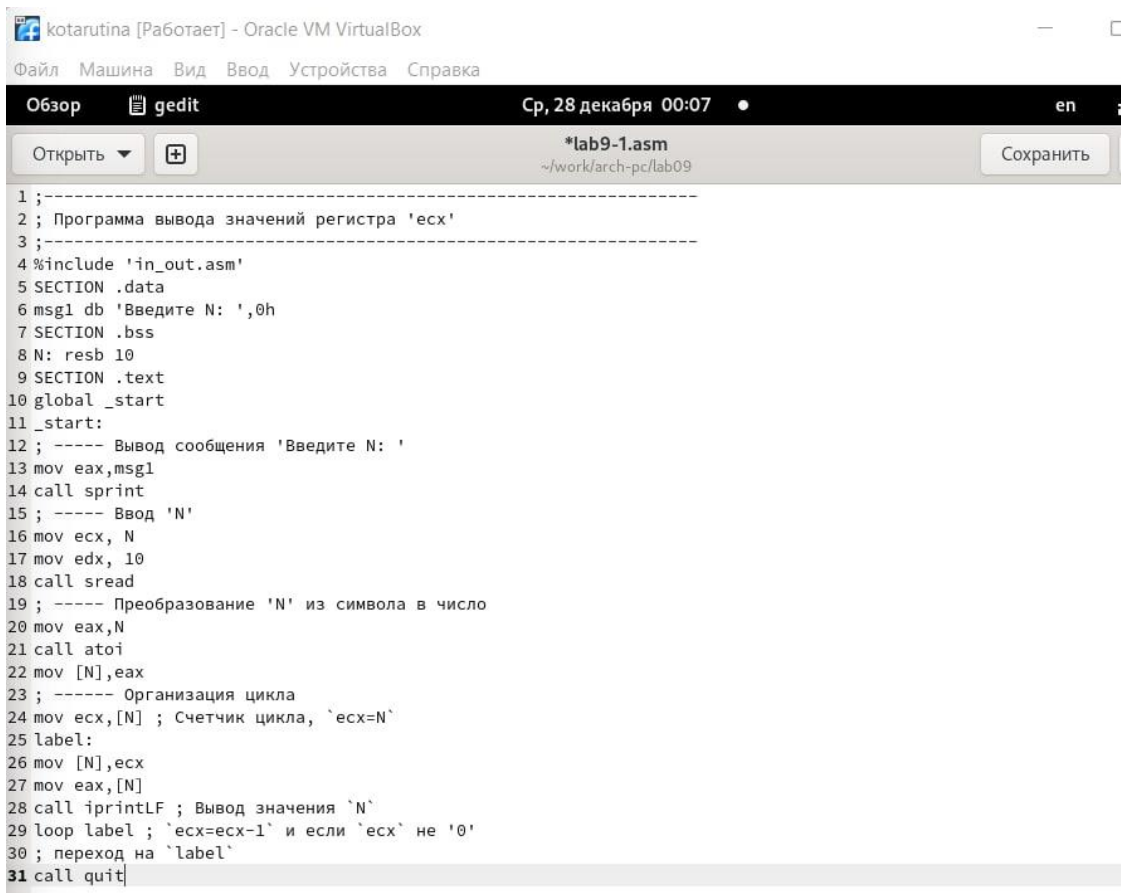
2 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы № 9, перехожу в него и создаю файл lab9-1.asm(рис. 1)

```
[kotarutina@fedora ~]$ mkdir ~/work/arch-pc/lab09
mkdir: невозможно создать каталог «/home/kotarutina/work/arch-pc/lab09»: Файл существует
[kotarutina@fedora ~]$ cd ~/work/arch-pc/lab09
[kotarutina@fedora lab09]$ touch lab9-1.asm
[kotarutina@fedora lab09]$
```

Рис. 1: Создание файла

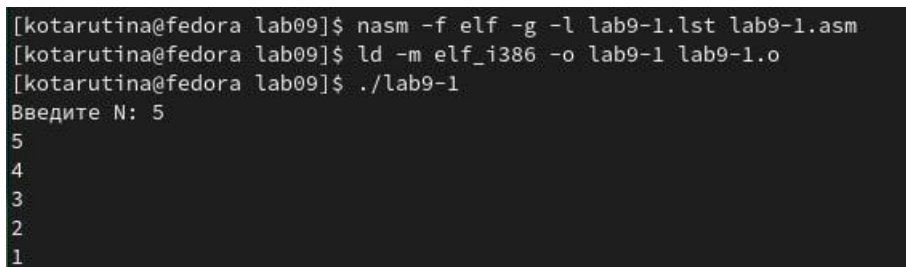
Ввожу в файл lab9-1.asm текст программы из листинга 9.1(рис. 2)



```
1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call iprintLF ; Вывод значения 'N'
29 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
30 ; переход на 'label'
31 call quit
```

Рис. 2: Текст программы

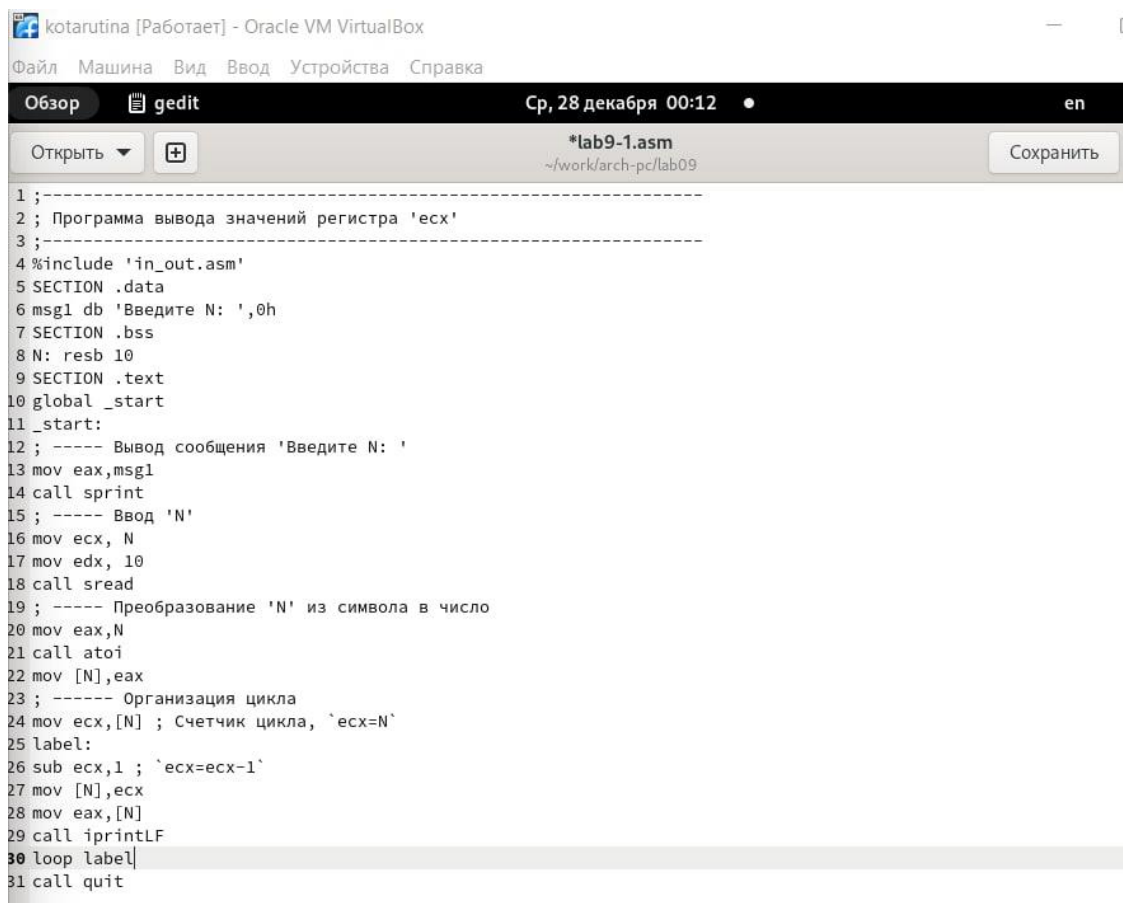
Создаю исполняемый файл и запускаю его.(рис. 3) Работа программы корректна, при введении числа 5 происходит 5 повторов цикла, в терминал выводится значения от 5 до 1 в убывающем порядке, соответствующее отсчёту оборотов цикла



```
[kotarutina@fedora lab09]$ nasm -f elf -g -l lab9-1.lst lab9-1.asm
[kotarutina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[kotarutina@fedora lab09]$ ./lab9-1
Введите N: 5
5
4
3
2
1
```

Рис. 3: Работа программы

Изменяю текст программы(рис. 4)



The screenshot shows a gedit editor window titled "kotarutina [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The toolbar has "Открыть", a "+" icon, and "Сохранить". The status bar shows "Ср, 28 декабря 00:12" and "en". The editor displays the following assembly code:

```
1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 sub ecx,1 ; `ecx=ecx-1`
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF
30 loop label
31 call quit
```

Рис. 4: Изменённый текст программы

Создаю исполняемый файл и проверяю его работу(рис. 5) Программа начинает работать не корректно, после отсчёта нуля(причём шаг не единица, а два), мы переходим на огромные значения

```
[kotarutina@fedora lab09]$ ./lab9-1
Введите N: 5
4
2
0
4294967294
4294967292
4294967290
4294967288
4294967286
4294967284
4294967282
4294967280
4294967278
4294967276
4294967274
4294967272
4294967270
4294967268
4294967266
4294967264
4294967262
4294967260
4294967258
```

Рис. 5: Работа программы

Снова изменяю текст программы(рис. 6)

```
1 ; вывод значений регистра 'ecx'
2
3
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25 label:
26 push ecx ; добавление значения ecx в стек
27 sub ecx,1
28 mov [N],ecx
29 mov eax,[N]
30 call iprintLF
31 pop ecx ; извлечение значения ecx из стека
32 loop label
33 call quit
```

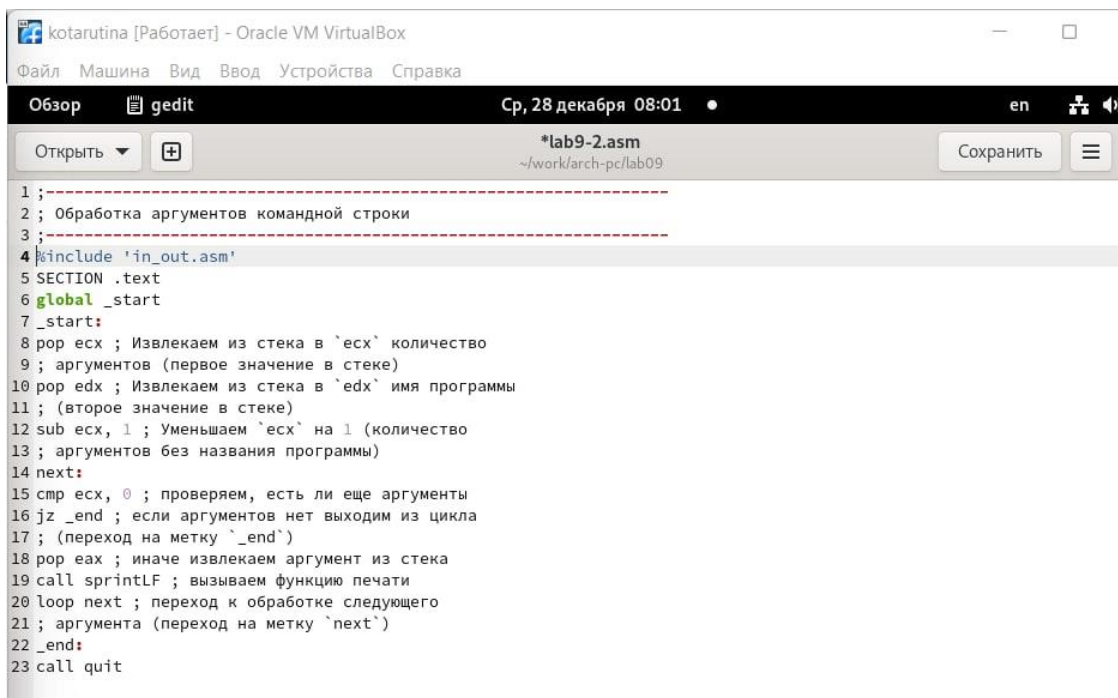
Рис. 6: Изменённый текст программы

Создаю исполняемый файл и проверяю его работу(рис. 7) Работа файла снова корректна, но в отличии от первой версии программы, счёт ведётся не от N-ного элемента и до 1, а от N - 1 и до 0

```
[kotarutina@fedora lab09]$ gedit lab9-1.asm
[kotarutina@fedora lab09]$ nasm -f elf -g -l lab9-1.lst lab9-1.asm
[kotarutina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[kotarutina@fedora lab09]$ ./lab9-1
Введите N: 5
4
3
2
1
0
```

Рис. 7: Работа программы

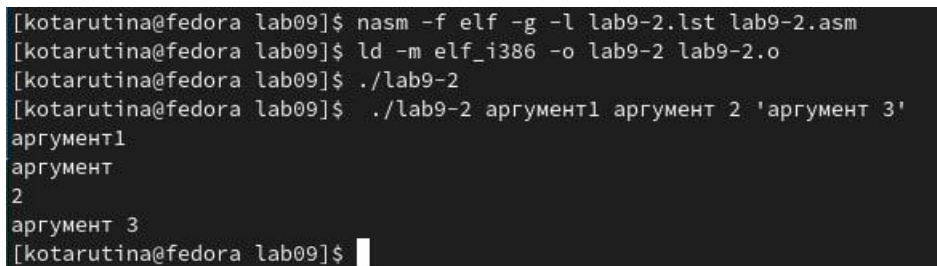
Создаю файл lab9-2.asm в каталоге ~/work/arch-pc/lab09. Внимательно изучаю текст программы из листинга 9.2 и ввожу в lab9-2.asm(рис. 8)



```
1 ;-----
2 ; обработка аргументов командной строки
3 ;-----
4 #include 'in_out.asm'
5 SECTION .text
6 global _start
7 _start:
8 pop ecx ; Извлекаем из стека в `ecx` количество
9 ; аргументов (первое значение в стеке)
10 pop edx ; Извлекаем из стека в `edx` имя программы
11 ; (второе значение в стеке)
12 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
13 ; аргументов без названия программы)
14 next:
15 cmp ecx, 0 ; проверяем, есть ли еще аргументы
16 jz _end ; если аргументов нет выходим из цикла
17 ; (переход на метку `_end`)
18 pop eax ; иначе извлекаем аргумент из стека
19 call printf ; вызываем функцию печати
20 loop next ; переход к обработке следующего
21 ; аргумента (переход на метку `next`)
22 _end:
23 call quit
```

Рис. 8: Текст программы

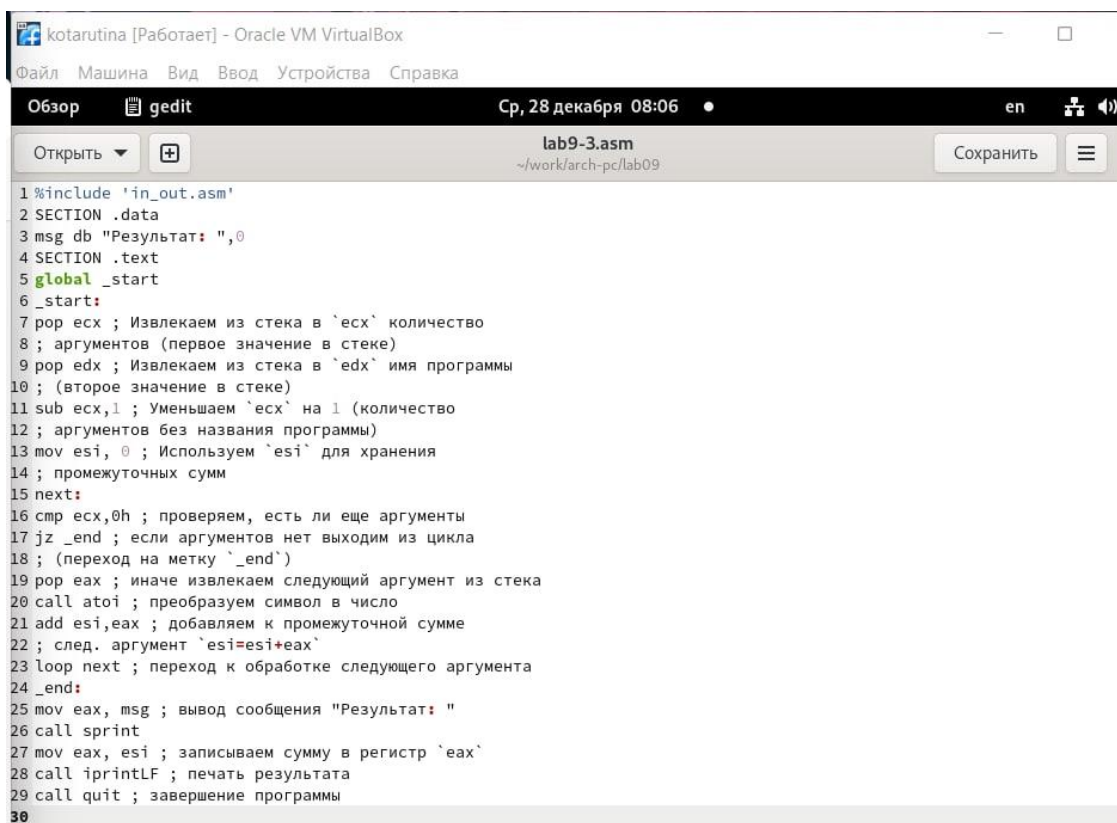
Создаю исполняемый файл и запускаю его. (рис. 9) В данном случае было обработано 4 аргумента, так как часть строки “аргумент 2” разделена пробелом и не имеет ” данных символов, а значит воспринимается как два разных аргумента



```
[kotarutina@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[kotarutina@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[kotarutina@fedora lab09]$ ./lab9-2
[kotarutina@fedora lab09]$ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[kotarutina@fedora lab09]$
```

Рис. 9: Работа программы

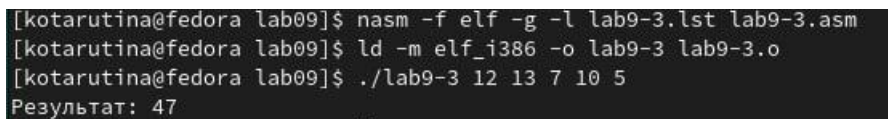
Создаю файл lab9-3.asm в каталоге ~/work/arch-pc/lab09. Внимательно изучаю текст программы из листинга 9.3 и ввожу в lab9-3.asm(рис. 10)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент `esi=esi+eax`
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintLF ; печать результата
29 call quit ; завершение программы
30
```

Рис. 10: Текст программы

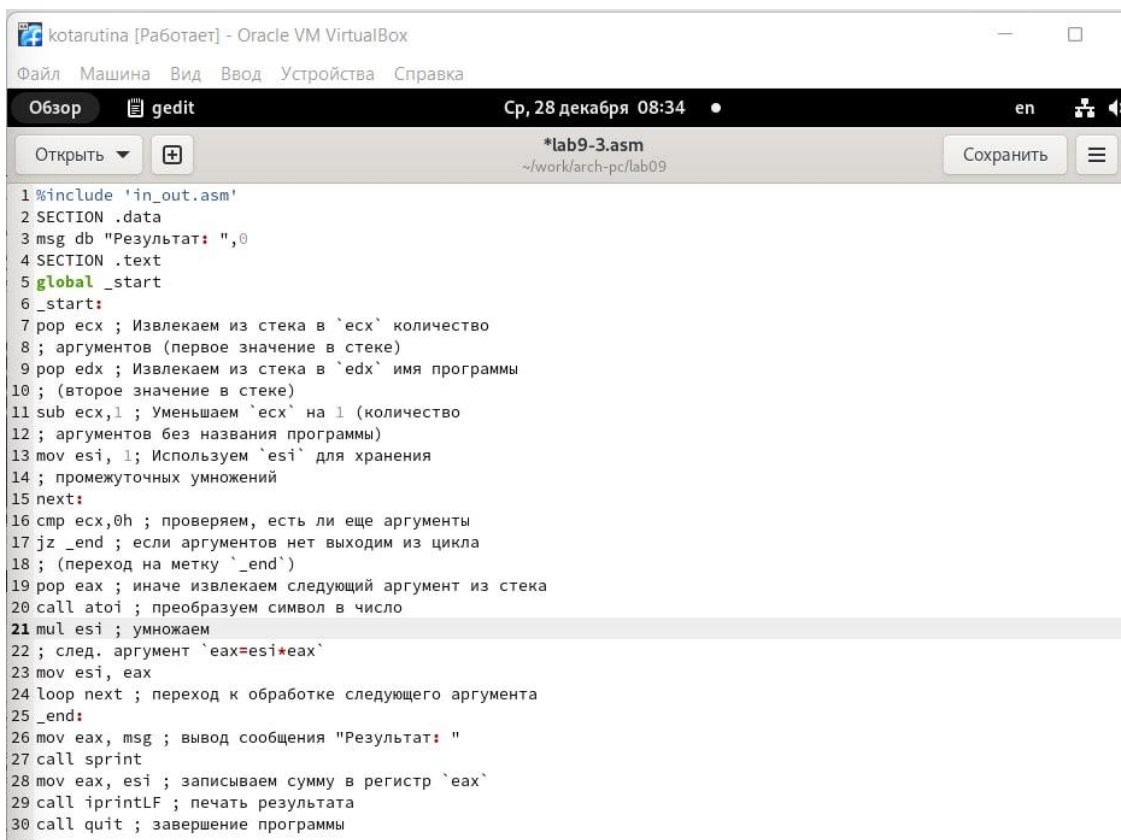
Создаю исполняемый файл и запускаю его. (рис. 11)



```
[kotarutina@fedora lab09]$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
[kotarutina@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[kotarutina@fedora lab09]$ ./lab9-3 12 13 7 10 5
Результат: 47
```

Рис. 11: Работа программы

Изменяю текст программы из листинга 9.3 для вычисления произведения аргументов командной строки. (рис. 12)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 1; Используем `esi` для хранения
14 ; промежуточных умножений
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 mul esi ; умножаем
22 ; след. аргумент `eax=esi*eax`
23 mov esi, eax
24 loop next ; переход к обработке следующего аргумента
25 _end:
26 mov eax, msg ; вывод сообщения "Результат: "
27 call sprint
28 mov eax, esi ; записываем сумму в регистр `eax`
29 call iprintLF ; печать результата
30 call quit ; завершение программы
```

Рис. 12: Текст изменённой программы

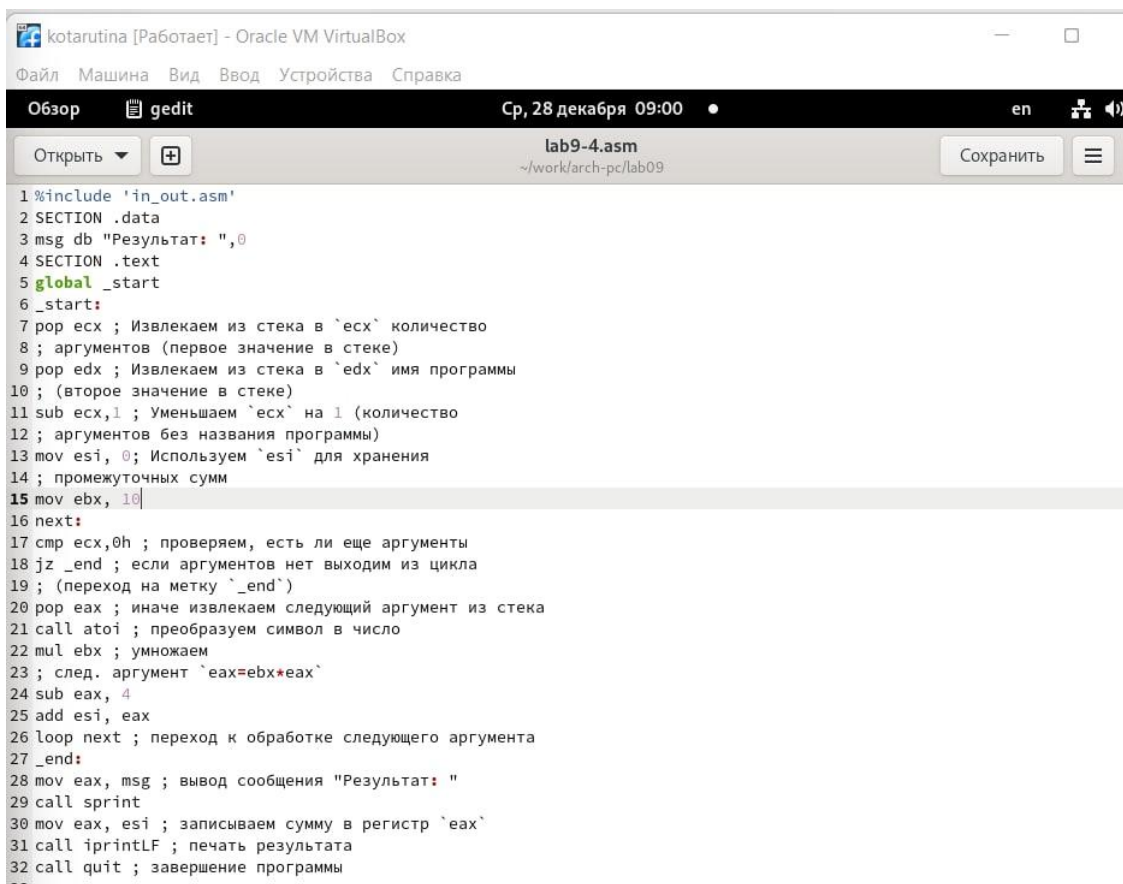
Создаю исполняемый файл и запускаю его(рис. 13)

```
[kotarutina@fedora lab09]$ gedit lab9-3.asm
[kotarutina@fedora lab09]$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
[kotarutina@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[kotarutina@fedora lab09]$ ./lab9-3 1 2 3 4 5
Результат: 120
```

Рис. 13: Работа файла

3 Выполнение самостоятельной работы

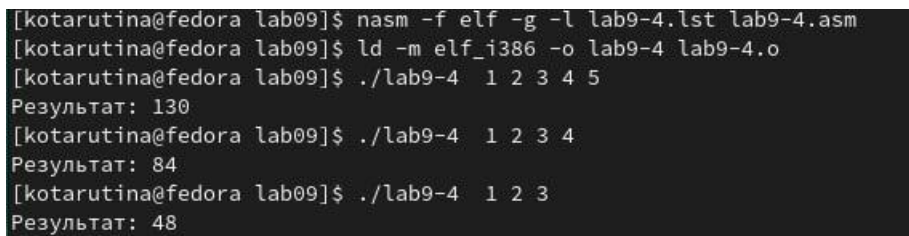
Пишу программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. (рис. 14) Вариант 9



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0; Используем `esi` для хранения
14 ; промежуточных сумм
15 mov ebx, 10
16 next:
17 cmp ecx,0h ; проверяем, есть ли еще аргументы
18 jz _end ; если аргументов нет выходим из цикла
19 ; (переход на метку `_end`)
20 pop eax ; иначе извлекаем следующий аргумент из стека
21 call atoi ; преобразуем символ в число
22 mul ebx ; умножаем
23 ; след. аргумент `eax=ebx*eax`
24 sub eax, 4
25 add esi, eax
26 loop next ; переход к обработке следующего аргумента
27 _end:
28 mov eax, msg ; вывод сообщения "Результат: "
29 call sprint
30 mov eax, esi ; записываем сумму в регистр `eax`
31 call iprintLF ; печать результата
32 call quit ; завершение программы
```

Рис. 14: Текст программы

Создаю программу по вычислению значений заданной функции для последовательности x (рис. 15)



```
[kotarutina@fedora lab09]$ nasm -f elf -g -l lab9-4.lst lab9-4.asm
[kotarutina@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o
[kotarutina@fedora lab09]$ ./lab9-4 1 2 3 4 5
Результат: 130
[kotarutina@fedora lab09]$ ./lab9-4 1 2 3 4
Результат: 84
[kotarutina@fedora lab09]$ ./lab9-4 1 2 3
Результат: 48
```

Рис. 15: Работа программы

4 Выводы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки прошло успешно