

Отчёт по лабораторной работе 8

Программирование цикла. Обработка аргументов командной строки.

Скобеева Алиса Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация циклов в NASM	6
2.2	Обработка аргументов командной строки	12
2.3	Задание для самостоятельной работы	17
3	Выводы	19

Список иллюстраций

2.1	Текст программы из файла lab8-1.asm	7
2.2	Результаты запуска программы из файла lab8-1.asm	8
2.3	Измененный текст программы	9
2.4	Результаты запуска измененной программы	10
2.5	Программа с использованием стека	11
2.6	Результаты запуска программы со стеком	12
2.7	Текст программы lab8-2.asm	13
2.8	Результаты запуска программы lab8-2.asm	13
2.9	Программа для вычисления суммы аргументов	14
2.10	Результаты вычисления суммы аргументов	15
2.11	Программа для вычисления произведения аргументов	16
2.12	Результаты вычисления произведения аргументов	16
2.13	Текст программы для вычисления суммы значений функции . . .	17
2.14	Результаты вычисления суммы значений функции	18

Список таблиц

1 Цель работы

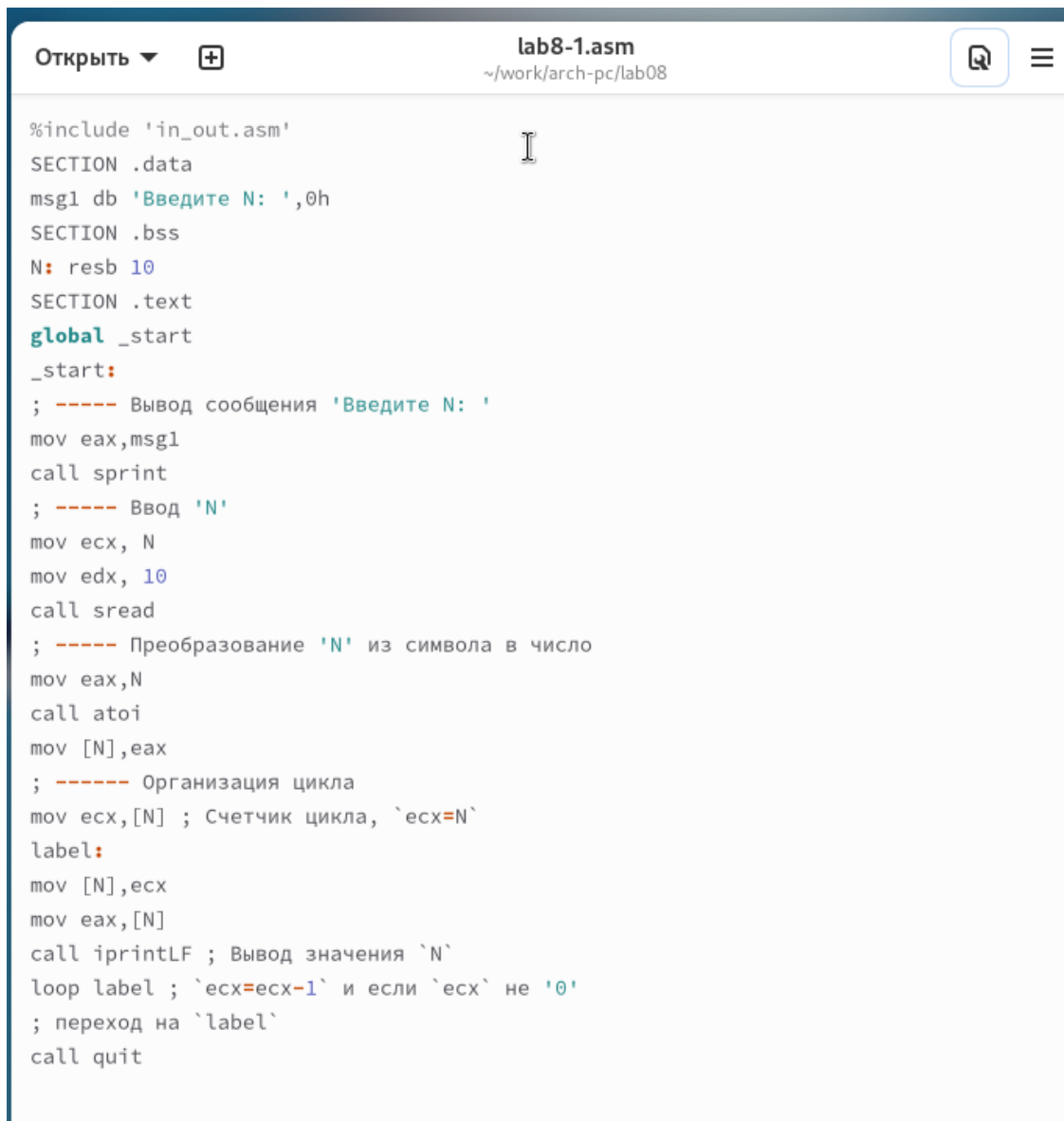
Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

2 Выполнение лабораторной работы

2.1 Реализация циклов в NASM

Мы создали каталог для программ лабораторной работы №8, перешли в него и добавили файл lab8-1.asm.

В файл lab8-1.asm был внесен текст программы из листинга 8.1. Затем мы скомпилировали и протестировали её выполнение.



```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Рис. 2.1: Текст программы из файла lab8-1.asm

```
aaskobeeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aaskobeeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
aaskobeeva@fedora:~/work/arch-pc/lab08$ □
```

Рис. 2.2: Результаты запуска программы из файла lab8-1.asm

На примере выяснили, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Мы изменили текст программы, добавив изменение значения регистра `ecx` в цикле, создали исполняемый файл и проверили его. При этом регистр `ecx` принимает значения, зависящие от начального `N`.

- Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`.

Открыть ▾ 

lab8-1.asm
~/work/arch-pc/lab08

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
; переход на `label`
call quit
```


Рис. 2.3: Измененный текст программы

```
4294886011
4294886009
4294886007
4294886005
4294886003
4294886001
4294885999
4294885997
^C
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
aaskobeeva@fedora:~/work/arch-pc/lab08$
```

Рис. 2.4: Результаты запуска измененной программы

Чтобы сохранить корректность работы программы при использовании регистра `ecx`, был применён стек. Мы добавили команды `push` и `pop` для сохранения значения счетчика цикла `loop`. Исполняемый файл был создан и протестирован. В этом случае число проходов цикла соответствует значению `N`, введенному с клавиатуры.

- Программа выводит числа от `N-1` до `0`, и цикл выполняется корректное количество раз.

Открыть ▾  lab8-1.asm
~/work/arch-pc/lab08

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
call quit
```

Рис. 2.5: Программа с использованием стека


```
aaskobeeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aaskobeeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 3
2
1
0
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
aaskobeeva@fedora:~/work/arch-pc/lab08$
```

Рис. 2.6: Результаты запуска программы со стеком

2.2 Обработка аргументов командной строки

В каталоге `~/work/arch-pc/lab08` мы создали файл `lab8-2.asm` и добавили в него текст программы из листинга 8.2. Программа была скомпилирована и запущена с аргументами.

- Программа обработала 4 аргумента.

Открыть ▾ 

lab8-2.asm
~/work/arch-pc/lab08

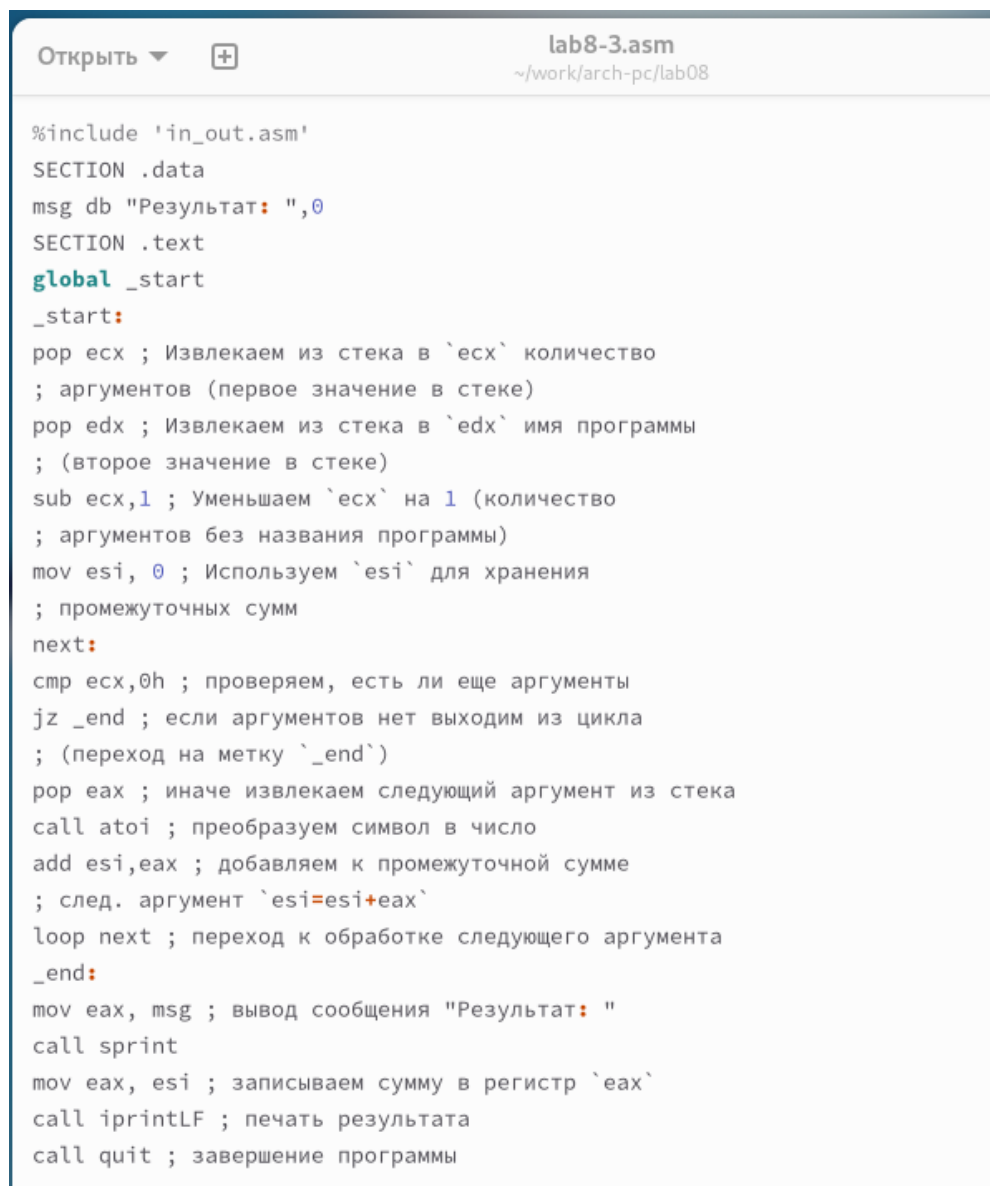
```
%include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем аргумент из стека
call sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```


Рис. 2.7: Текст программы lab8-2.asm

```
aaskobeeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aaskobeeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-2
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
aaskobeeva@fedora:~/work/arch-pc/lab08$
```

Рис. 2.8: Результаты запуска программы lab8-2.asm

Мы изучили пример программы, которая вычисляет сумму чисел, переданных в качестве аргументов.



```
Открыть ▾  lab8-3.asm
~\work\arch-pc\lab08

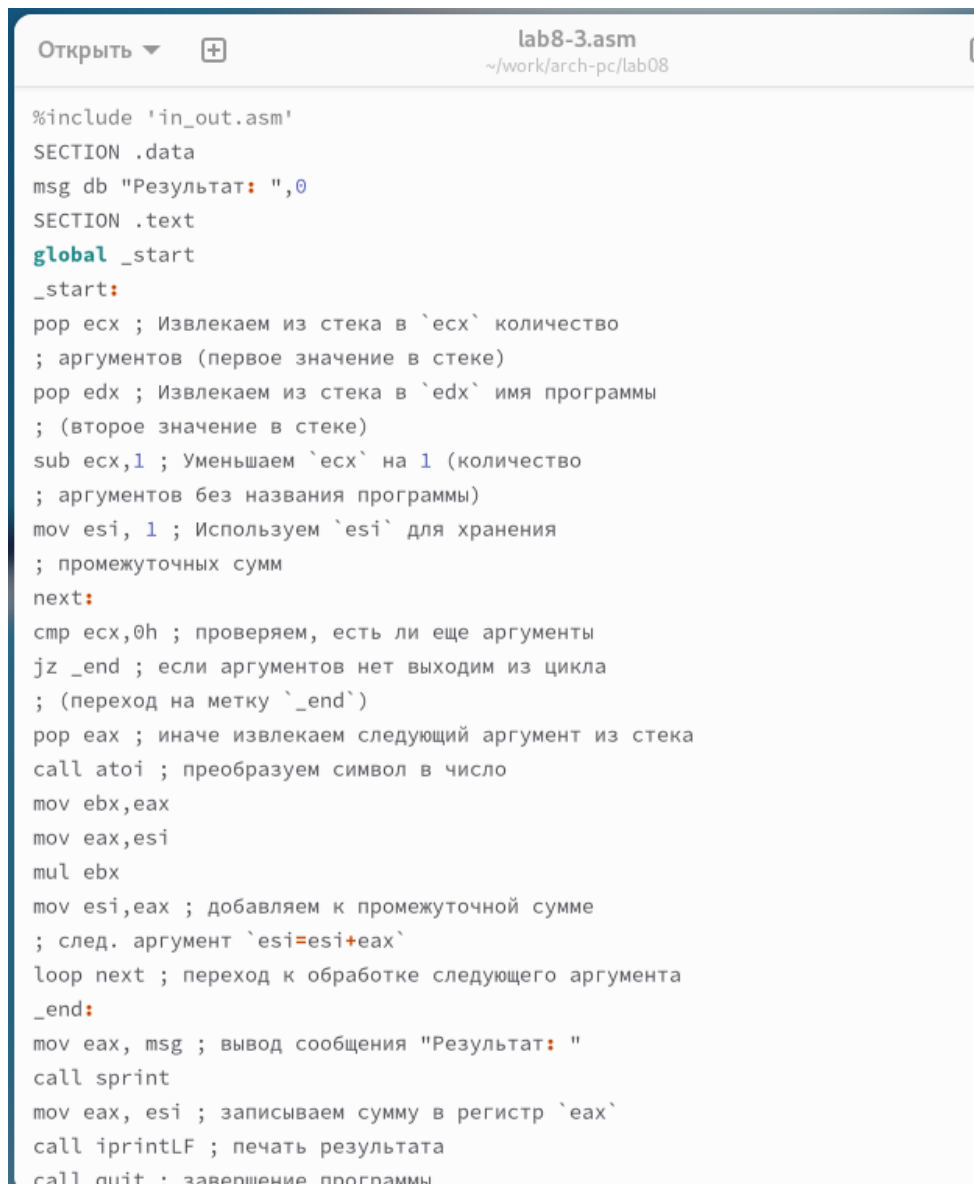
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 2.9: Программа для вычисления суммы аргументов

```
aaskobeeva@fedora:~/work/arch-pc/lab08$  
aaskobeeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm  
aaskobeeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3  
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-3  
Результат: 0  
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-3 4 5 6  
Результат: 15  
aaskobeeva@fedora:~/work/arch-pc/lab08$
```

Рис. 2.10: Результаты вычисления суммы аргументов

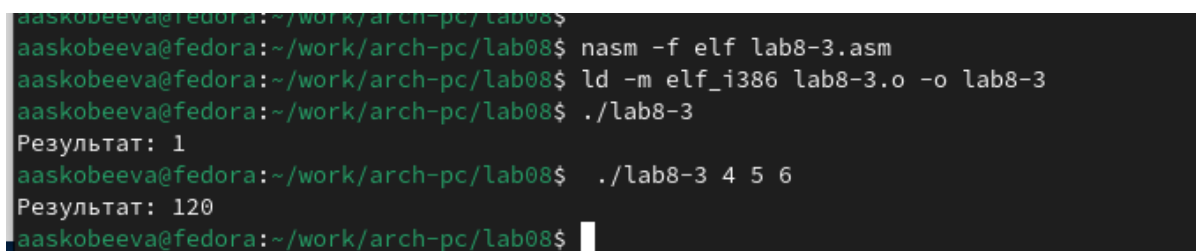
Изменили текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.



```
Открыть ▾ + lab8-3.asm
~/work/arch-pc/lab08

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mov ebx,eax
mov eax,esi
mul ebx
mov esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 2.11: Программа для вычисления произведения аргументов



```
aaskobeeva@fedora:~/work/arch-pc/lab08$
aaskobeeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aaskobeeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-3
Результат: 1
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./lab8-3 4 5 6
Результат: 120
aaskobeeva@fedora:~/work/arch-pc/lab08$
```

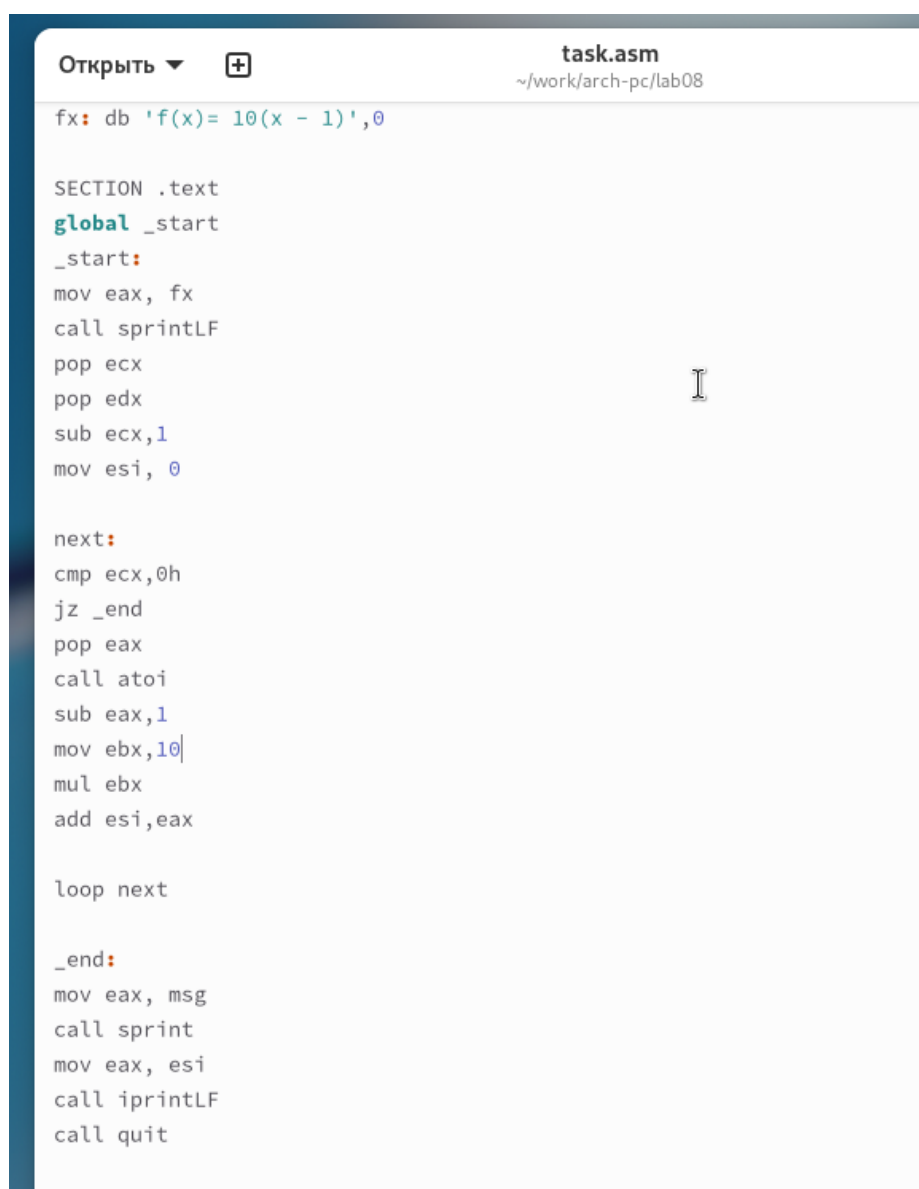
Рис. 2.12: Результаты вычисления произведения аргументов

2.3 Задание для самостоятельной работы

Мы разработали программу для вычисления суммы значений функции $f(x)$ для набора $x = x_1, x_2, \dots, x_n$. Вид функции $f(x)$ был выбран в соответствии с вариантом задания №12 из таблицы 8.1:

- Для варианта 17: $f(x) = 10(x - 1)$.

Программа была протестирована на нескольких наборах x .



```
task.asm
~/work/arch-pc/lab08

fx: db 'f(x)= 10(x - 1)',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintf
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
sub eax,1
mov ebx,10
mul ebx
add esi,eax

loop next

_end:
mov eax, msg
call sprintf
mov eax, esi
call iprintLF
call quit
```

Рис. 2.13: Текст программы для вычисления суммы значений функции

```
aaskobeeva@fedora:~/work/arch-pc/lab08$ nasm -f elf task.asm
aaskobeeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 task.o -o task
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./task
f(x)= 10(x - 1)
Результат: 0
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./task 2
f(x)= 10(x - 1)
Результат: 10
aaskobeeva@fedora:~/work/arch-pc/lab08$ ./task 2 3 4 5 6
f(x)= 10(x - 1)
Результат: 150
aaskobeeva@fedora:~/work/arch-pc/lab08$
```

Рис. 2.14: Результаты вычисления суммы значений функции

3 Выводы

Мы освоили работу со стеком, циклами и аргументами на ассемблере NASM.