

Лабораторная работа 8

Программирование цикла. Обработка аргументов командной строки.

Глушенок Анна Александровна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Задание 1. Реализация циклов в NASM	6
2.2	Задание 2. Обработка аргументов командной строки	10
3	Задания для самостоятельной работы	14
4	Вывод	17

Список иллюстраций

2.1	Создание рабочего каталога	6
2.2	Программа вывода значений регистра есх	7
2.3	Проверка работы программы	7
2.4	Внесение изменений в программу	8
2.5	Результат работы программы	8
2.6	Внесение изменений в программу	9
2.7	Результат работы программы	10
2.8	Вывод аргументов командной строки	11
2.9	Результат работы программы	11
2.10	Сумма аргументов командной строки	12
2.11	Результат работы программы	12
2.12	Произведение аргументов командной стр	13
2.13	Результат работы программы	13
3.1	Вычисление суммы значений функции	15
3.2	Результат работы программы	16

Список таблиц

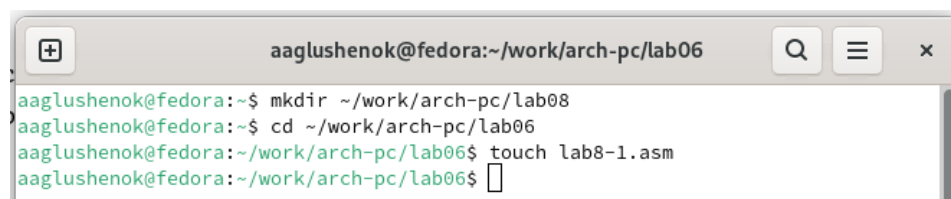
1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Задание 1. Реализация циклов в NASM

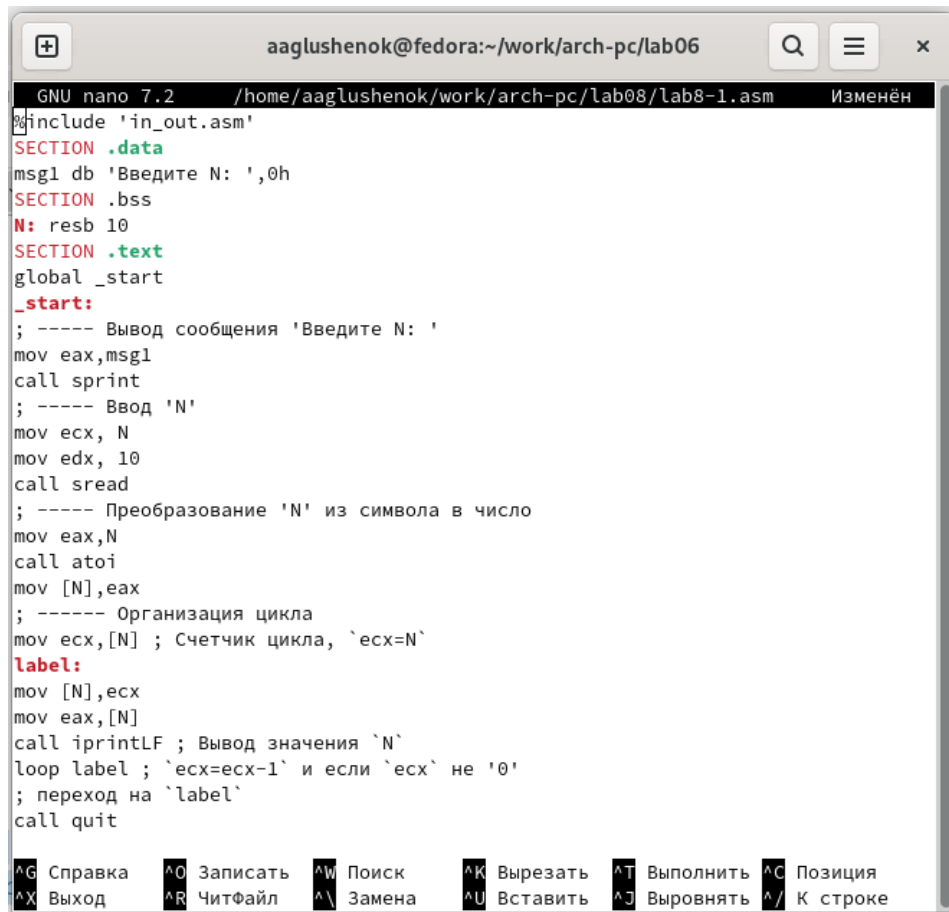
1. Создаем каталог для программ лабораторной работы № 8, переходим в него и создаем файл lab8-1.asm.



```
aaglushenok@fedora:~/work/arch-pc/lab06
aaglushenok@fedora:~$ mkdir ~/work/arch-pc/lab08
aaglushenok@fedora:~$ cd ~/work/arch-pc/lab06
aaglushenok@fedora:~/work/arch-pc/lab06$ touch lab8-1.asm
aaglushenok@fedora:~/work/arch-pc/lab06$
```

Рис. 2.1: Создание рабочего каталога

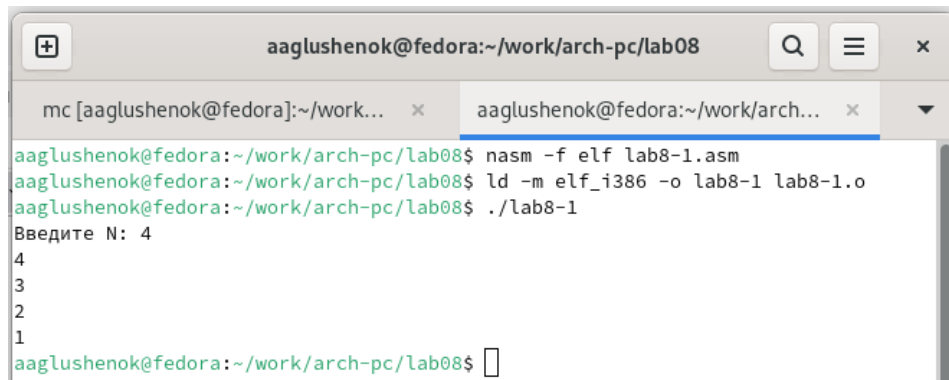
2. Вводим в файл lab8-1.asm текст программы из листинга 8.1.(Программа вывода значений регистра есх). Создаем исполняемый файл и проверяем его работу.



```
GNU nano 7.2 /home/aaglushmanok/work/arch-pc/lab08/lab8-1.asm
%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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выровнять ^/ К строке

Рис. 2.2: Программа вывода значений регистра ecx

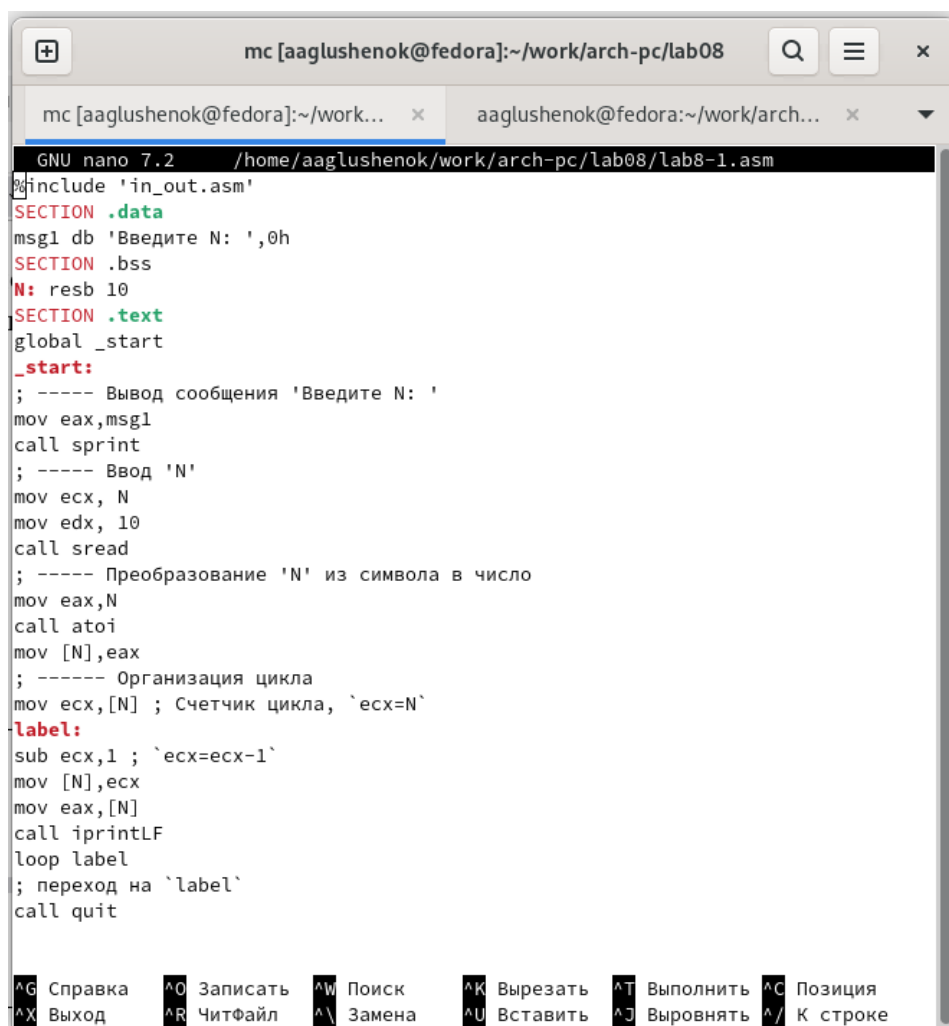


```
aaglushmanok@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aaglushmanok@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aaglushmanok@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
aaglushmanok@fedora:~/work/arch-pc/lab08$
```

Рис. 2.3: Проверка работы программы

3. Меняем текст программы, добавив изменение значение регистра ecx в цикле. Создаем исполняемый файл и проверяем его работу. Регистр eax

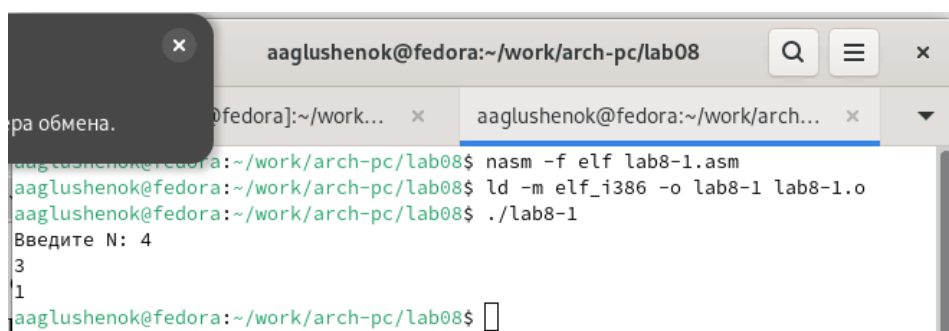
принимает значения 3, 1; число проходов цикла не соответствует значению N.



```
GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab08/lab8-1.asm
%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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Вывернуть ^_ К строке

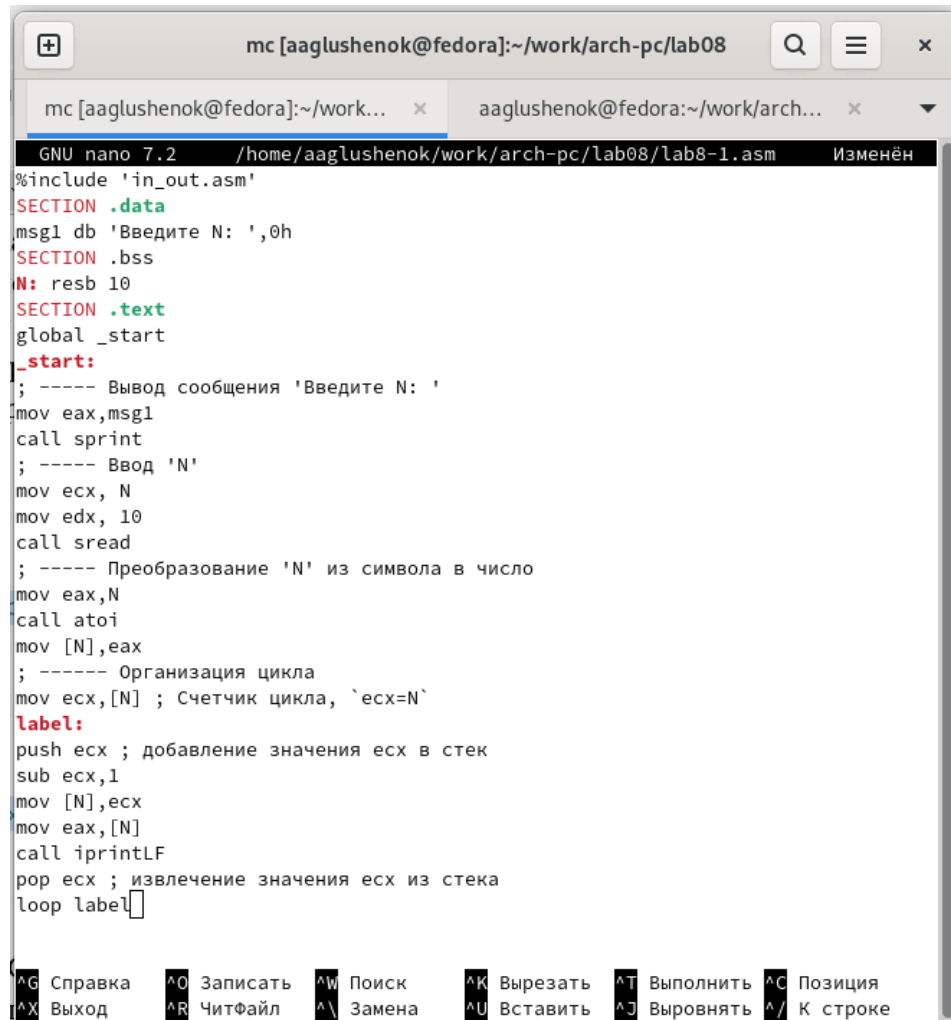
Рис. 2.4: Внесение изменений в программу



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

Рис. 2.5: Результат работы программы

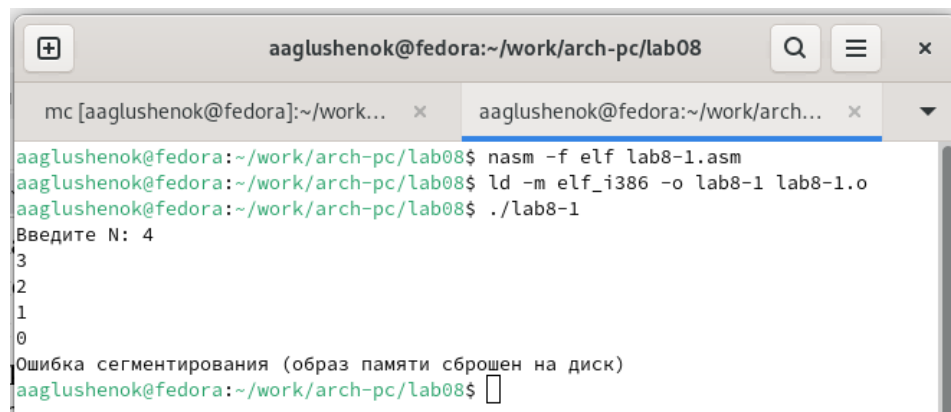
4. Вносим изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. Тогда число проходов цикла становится равным числу N.



```
mc [aaglushenok@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab08/lab8-1.asm
%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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 2.6: Внесение изменений в программу

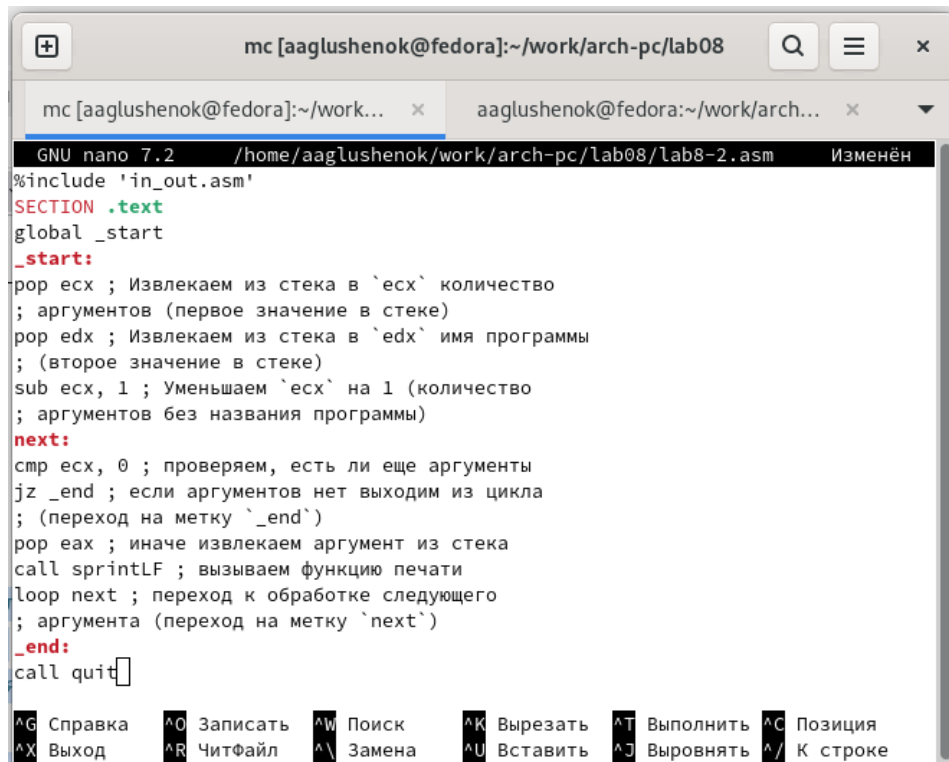


```
aaglushmanok@fedora:~/work/arch-pc/lab08
mc [aaglushmanok@fedora]:~/work... x aaglushmanok@fedora:~/work/arch... x
aaglushmanok@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aaglushmanok@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aaglushmanok@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
Ошибка сегментирования (образ памяти сброшен на диск)
aaglushmanok@fedora:~/work/arch-pc/lab08$
```

Рис. 2.7: Результат работы программы

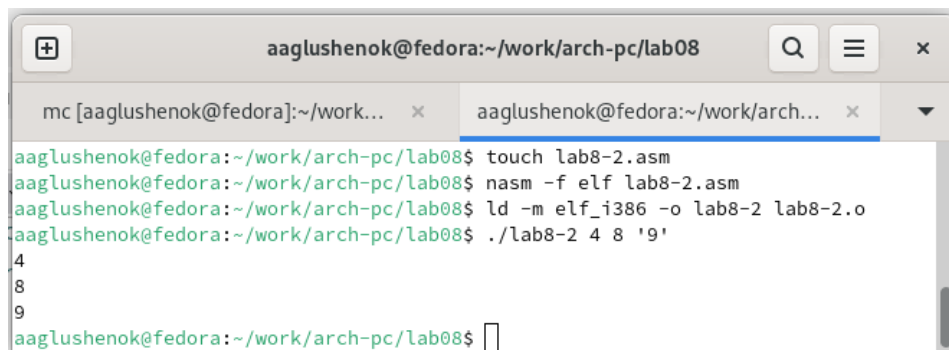
2.2 Задание 2. Обработка аргументов командной строки

4. Создаем файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и вводим в него текст программы из листинга 8.2. (Программа выводящая на экран аргументы командной строки). Создаем исполняемый файл и запускаем его, указав аргументы из текста лабораторной работы.



```
mc [aaglushenok@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab08/lab8-2.asm
%include 'in_out.asm'
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por 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`)
por 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
```

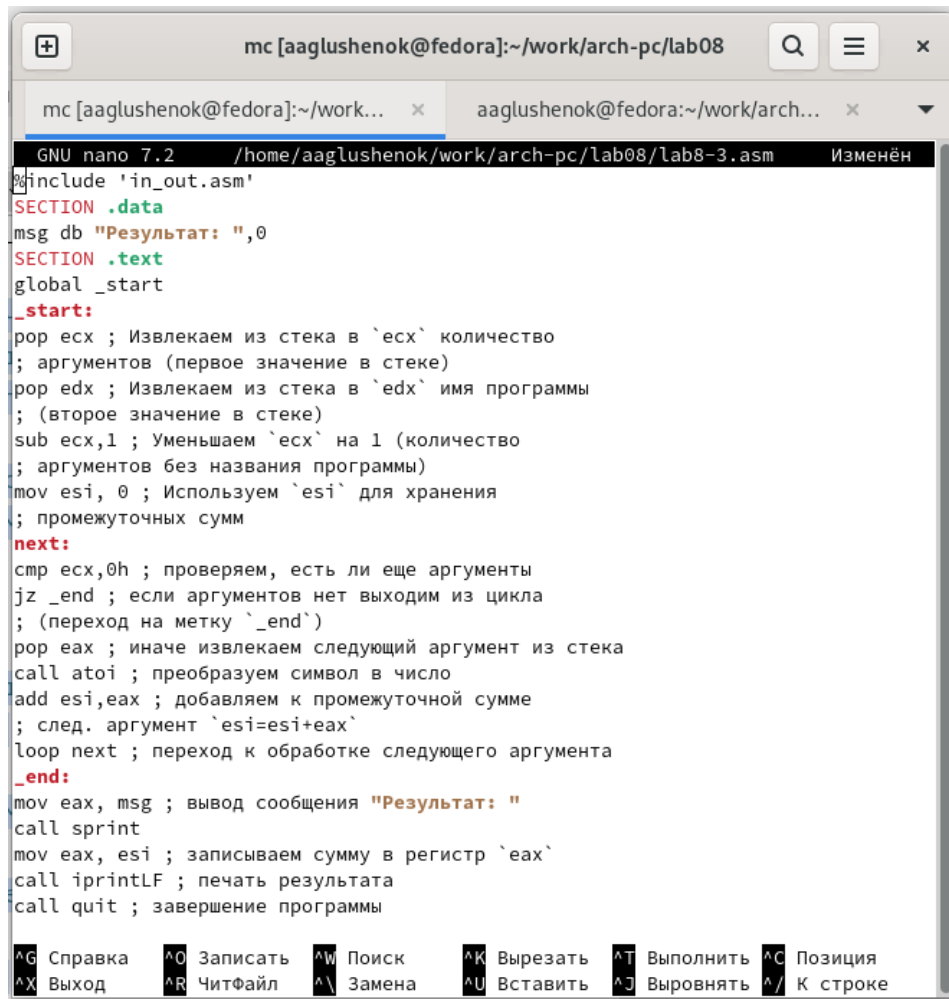
Рис. 2.8: Вывод аргументов командной строки



```
aaglushenok@fedora:~/work/arch-pc/lab08
mc [aaglushenok@fedora]:~/work/arch-pc/lab08
aaglushenok@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
aaglushenok@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aaglushenok@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aaglushenok@fedora:~/work/arch-pc/lab08$ ./lab8-2 4 8 '9'
4
8
9
aaglushenok@fedora:~/work/arch-pc/lab08$
```

Рис. 2.9: Результат работы программы

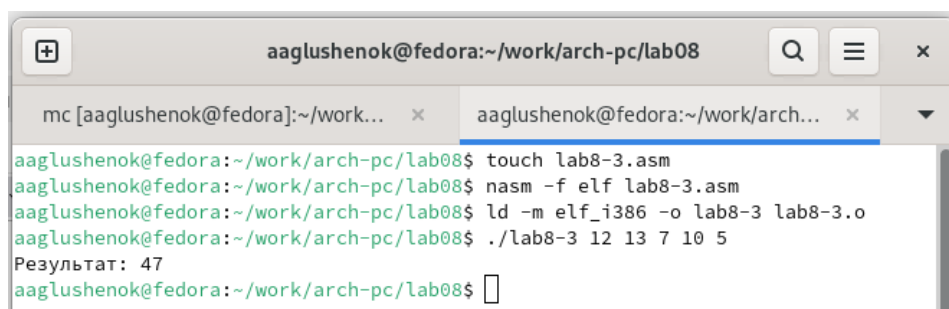
5. Создаем файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и вводим в него текст программы из листинга 8.3. (Программа вычисления суммы аргументов командной строки). Создаем исполняемый файл и запускаем его, указав аргументы.



```
mc [aaglushenok@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab08/lab8-3.asm
#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 sprintf
    mov eax, esi ; записываем сумму в регистр `eax`
    call iprintLF ; печать результата
    call quit ; завершение программы
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/_ К строке

Рис. 2.10: Сумма аргументов командной строки



```
aaglushenok@fedora:~/work/arch-pc/lab08
mc [aaglushenok@fedora]:~/work/arch-pc/lab08$ touch lab8-3.asm
aaglushenok@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aaglushenok@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aaglushenok@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
aaglushenok@fedora:~/work/arch-pc/lab08$
```

Рис. 2.11: Результат работы программы

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



```
mc [aaglushenok@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab08/lab8-3.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx          ; Извлекаем из стека в ecx количество
    pop edx          ; Извлекаем из стека в edx имя программы
    sub ecx, 1       ; Уменьшаем ecx на 1 (количество аргументов без названия)

    mov esi, 1       ; Используем esi для хранения произведения, начинаем с 1
next:
    cmp ecx, 0h      ; проверяем, есть ли еще аргументы
    jz _end          ; если аргументов нет выходим из цикла

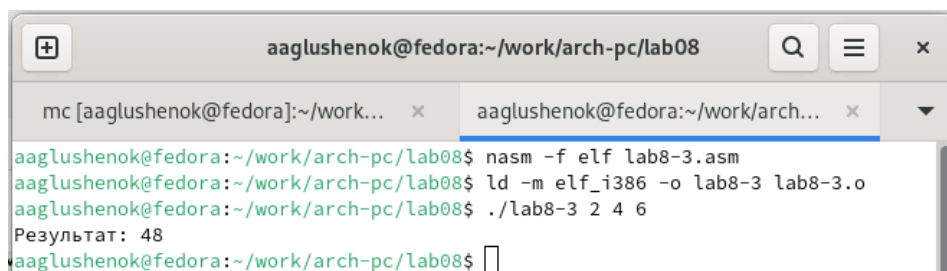
    pop eax          ; иначе извлекаем следующий аргумент из стека
    call atoi        ; преобразуем символ в число
    mul esi          ; умножаем текущее произведение на полученное число
    mov esi, eax      ; сохраняем результат обратно в esi

    loop next        ; переходим к обработке следующего аргумента

_end:
    mov eax, msg      ; вывод сообщения "Результат: "
    call sprint
    mov eax, esi      ; записываем произведение в регистр eax
    call iprintLF     ; печать результата
    call quit         ; завершение программы
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 2.12: Произведение аргументов командной стр

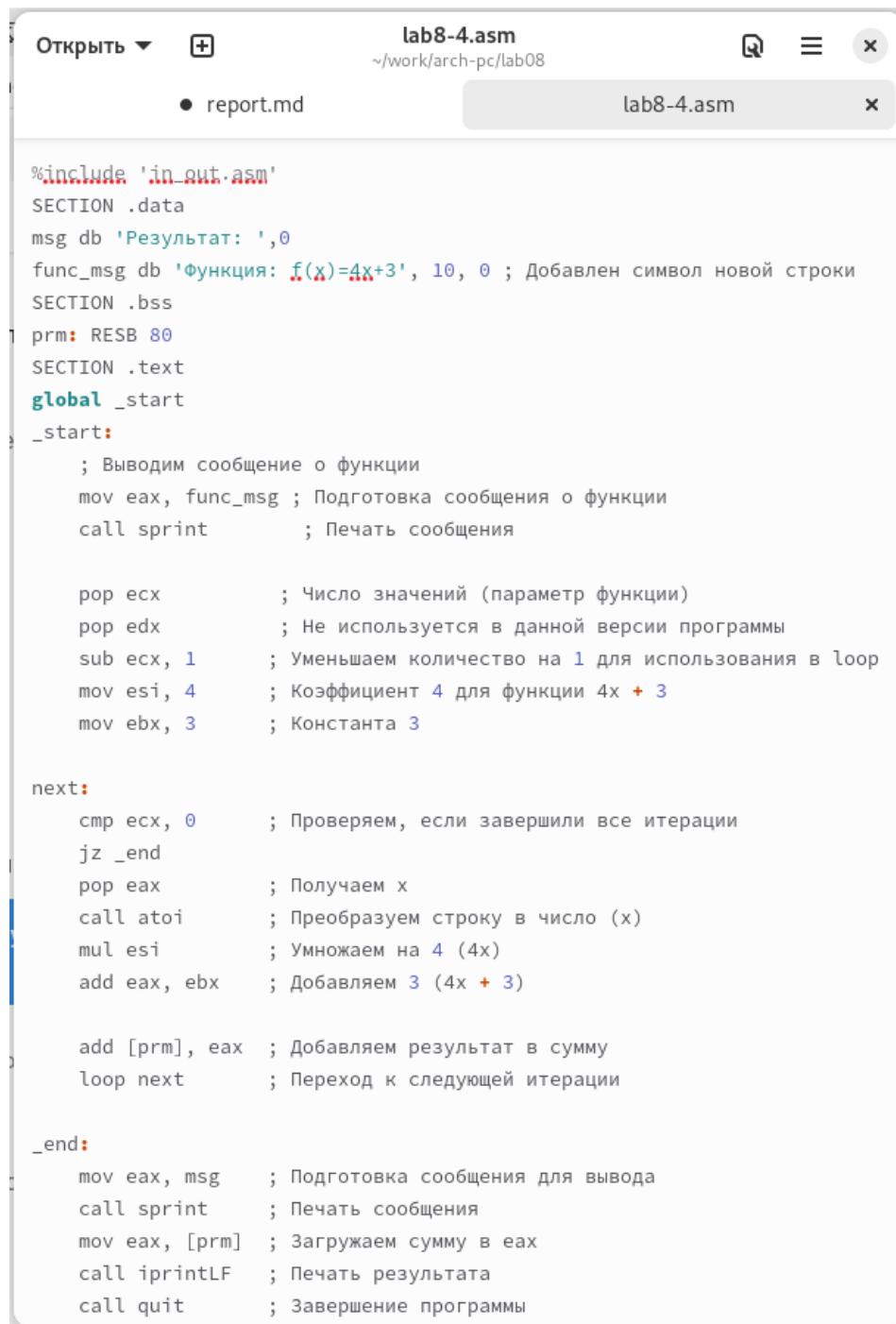


```
aaglushenok@fedora:~/work/arch-pc/lab08
mc [aaglushenok@fedora]:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aaglushenok@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aaglushenok@fedora:~/work/arch-pc/lab08$ ./lab8-3 2 4 6
Результат: 48
aaglushenok@fedora:~/work/arch-pc/lab08$
```

Рис. 2.13: Результат работы программы

3 Задания для самостоятельной работы

1. Создаем программу, которая находит сумму значений функции $f(x)$ от всех введенных пользователем аргументов x . Создаем исполняемый файл и проверяем его работу на нескольких наборах аргументов.



```
%include 'in_out.asm'
SECTION .data
msg db 'Результат: ',0
func_msg db 'Функция: f(x)=4x+3', 10, 0 ; Добавлен символ новой строки
SECTION .bss
prm: RESB 80
SECTION .text
global _start
_start:
    ; Выводим сообщение о функции
    mov eax, func_msg ; Подготовка сообщения о функции
    call sprint        ; Печать сообщения

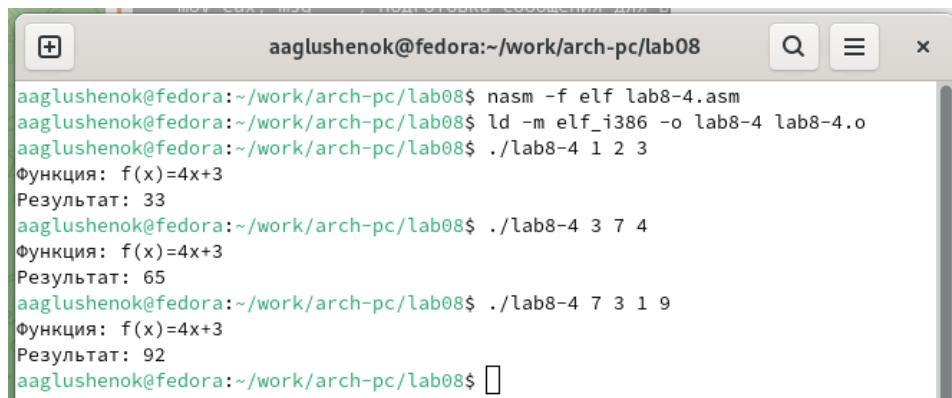
    pop ecx            ; Число значений (параметр функции)
    pop edx            ; Не используется в данной версии программы
    sub ecx, 1         ; Уменьшаем количество на 1 для использования в loop
    mov esi, 4         ; Коэффициент 4 для функции 4x + 3
    mov ebx, 3         ; Константа 3

next:
    cmp ecx, 0         ; Проверяем, если завершили все итерации
    jz _end
    pop eax             ; Получаем x
    call atoi          ; Преобразуем строку в число (x)
    mul esi             ; Умножаем на 4 (4x)
    add eax, ebx        ; Добавляем 3 (4x + 3)

    add [prm], eax      ; Добавляем результат в сумму
    loop next          ; Переход к следующей итерации

_end:
    mov eax, msg        ; Подготовка сообщения для вывода
    call sprint         ; Печать сообщения
    mov eax, [prm]      ; Загружаем сумму в eax
    call iprintLF       ; Печать результата
    call quit           ; Завершение программы
```

Рис. 3.1: Вычисление суммы значений функции



```
aaglushmanok@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aaglushmanok@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aaglushmanok@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция:  $f(x)=4x+3$ 
Результат: 33
aaglushmanok@fedora:~/work/arch-pc/lab08$ ./lab8-4 3 7 4
Функция:  $f(x)=4x+3$ 
Результат: 65
aaglushmanok@fedora:~/work/arch-pc/lab08$ ./lab8-4 7 3 1 9
Функция:  $f(x)=4x+3$ 
Результат: 92
aaglushmanok@fedora:~/work/arch-pc/lab08$
```

Рис. 3.2: Результат работы программы

4 Вывод

В ходе выполнения лабораторной работы мне удалось приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки