

# **Отчёт по лабораторной работе №8**

Ярослав Антонович Меркулов

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
2.1	Реализация циклов в NASM . . . . .	5
2.2	Обработка аргументов командной строки . . . . .	9
<b>3</b>	<b>Выполнение самостоятельной работы</b>	<b>13</b>
<b>4</b>	<b>Выводы</b>	<b>15</b>

# Список иллюстраций

2.1	Создание каталога, переход в него, создание файла . . . . .	5
2.2	Введённый текст программы . . . . .	6
2.3	Работа программы . . . . .	6
2.4	Изменённый файл . . . . .	7
2.5	Работа новой программы . . . . .	7
2.6	Программа со стеком . . . . .	8
2.7	Работа программы . . . . .	8
2.8	Файл lab8-2.asm . . . . .	9
2.9	Работа программы . . . . .	10
2.10	Файл lab8-3.asm . . . . .	10
2.11	Работа программы . . . . .	11
2.12	Программа для вычисления произведения . . . . .	11
2.13	Работа программы . . . . .	12
3.1	Готовый lab7-3.asm . . . . .	13
3.2	Работа программы . . . . .	14

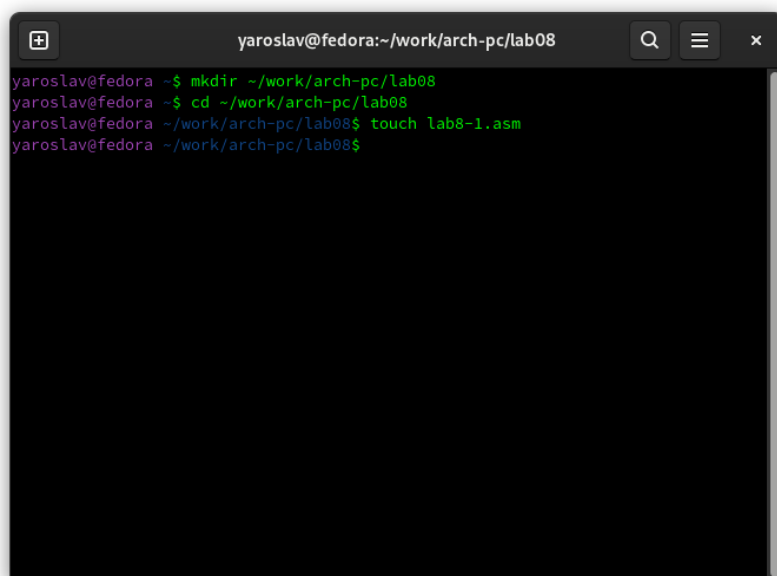
# 1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

## 2 Выполнение лабораторной работы

### 2.1 Реализация циклов в NASM

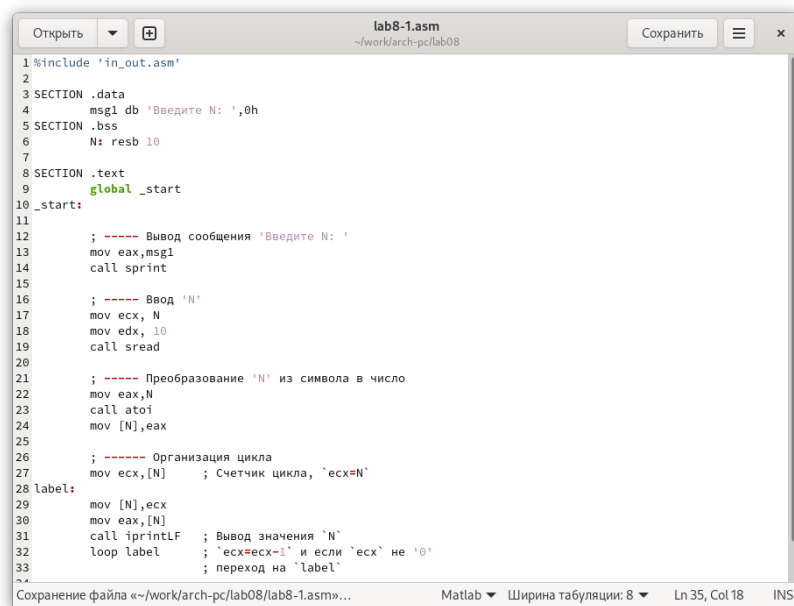
1. Создаём каталог для лабораторной работы, переходим в него и создаём файл lab8-1.asm.



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

Рис. 2.1: Создание каталога, переход в него, создание файла

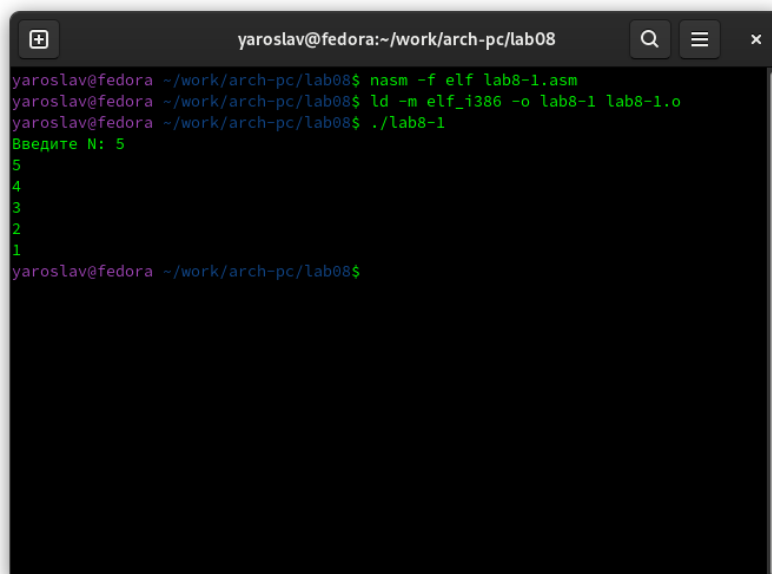
2. Вводим текст программы из листинга 8.1.



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

Рис. 2.2: Введенный текст программы

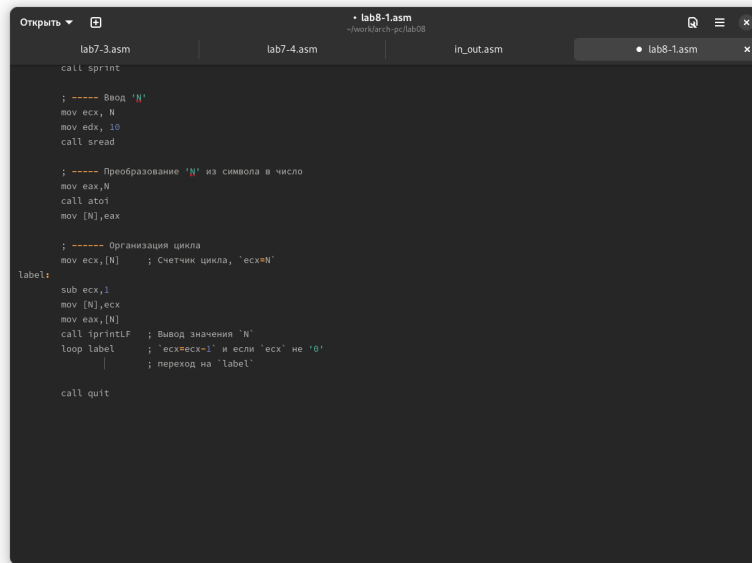
3. Создаём исполняемый файл и запускаем.



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

Рис. 2.3: Работа программы

4. Изменяем текст программы, создаём исполняемый файл и запускаем. Программа перескакивает через число (некорректная работа).



```
call sprint

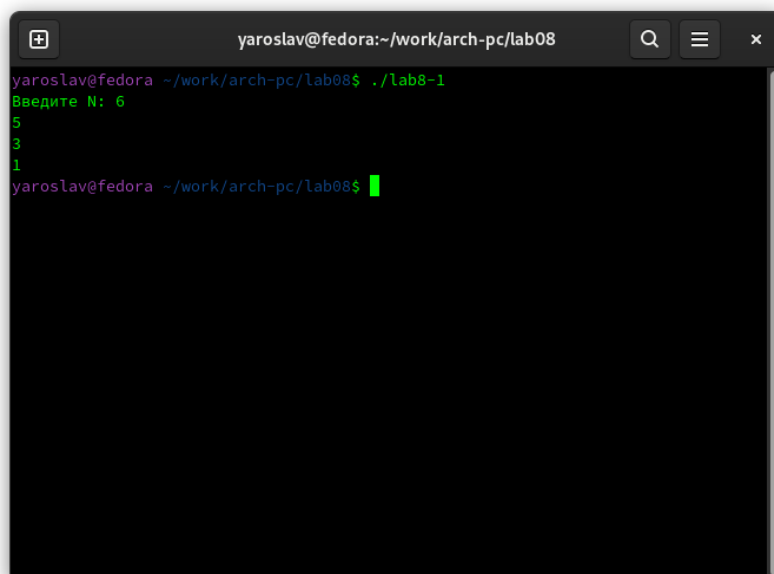
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call read

; ----- 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
mov [N], ecx
mov eax, [N]
call printf ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
           ; переход на 'label'

call quit
```

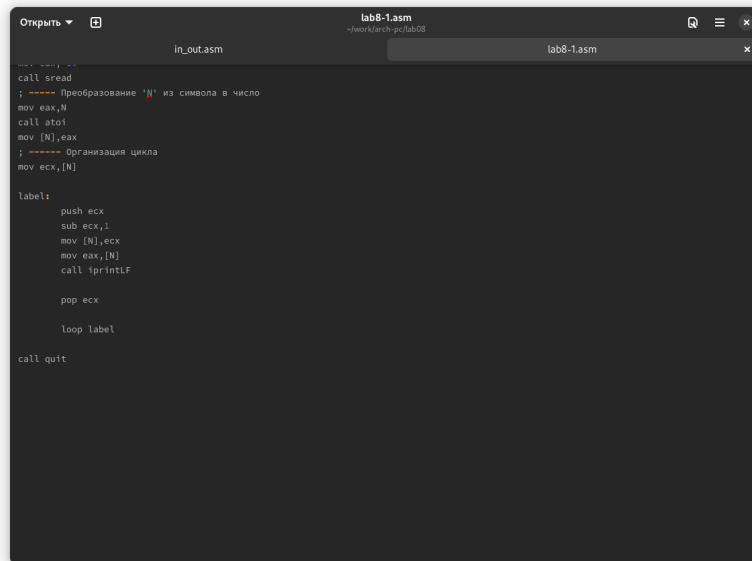
Рис. 2.4: Изменённый файл



```
yaroslav@fedora: ~/work/arch-pc/lab08
yaroslav@fedora ~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
yaroslav@fedora ~/work/arch-pc/lab08$
```

Рис. 2.5: Работа новой программы

5. Меняем программу так, чтоб использовался стек.



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

call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N]

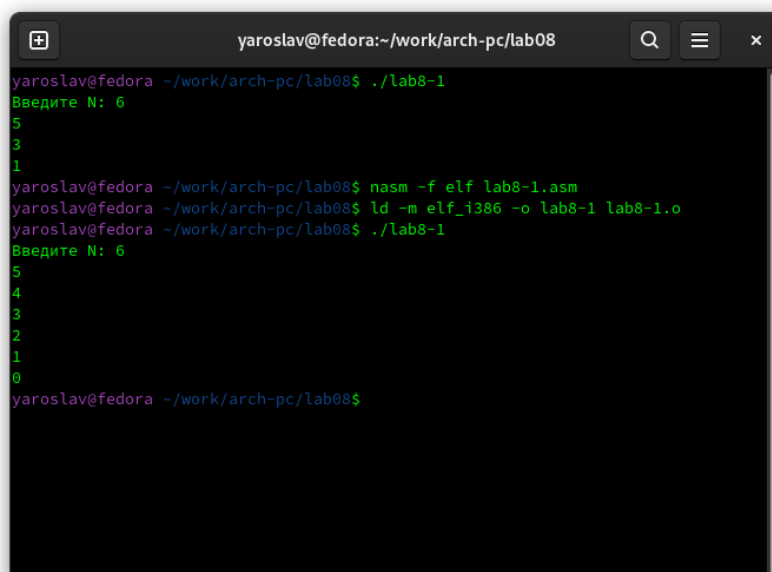
label:
    push ecx
    sub ecx,1
    mov [N],ecx
    mov eax,[N]
    call iprintfLF

    pop ecx
    loop label

call quit
```

Рис. 2.6: Программа со стеком

6. Создаём исполняемый файл и запускаем.



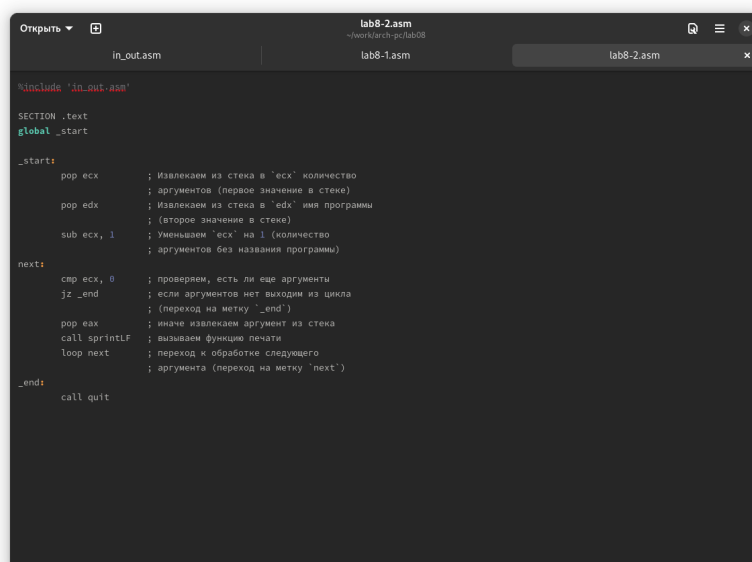
```
yaroslav@fedora:~/work/arch-pc/lab08
yaroslav@fedora ~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
yaroslav@fedora ~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
yaroslav@fedora ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
yaroslav@fedora ~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
4
3
2
1
0
yaroslav@fedora ~/work/arch-pc/lab08$
```

Рис. 2.7: Работа программы



## 2.2 Обработка аргументов командной строки

7. Создаём файл lab8-2.asm и записываем туда листинг 8-2.



```
#include 'in_out.asm'

SECTION .text
global _start

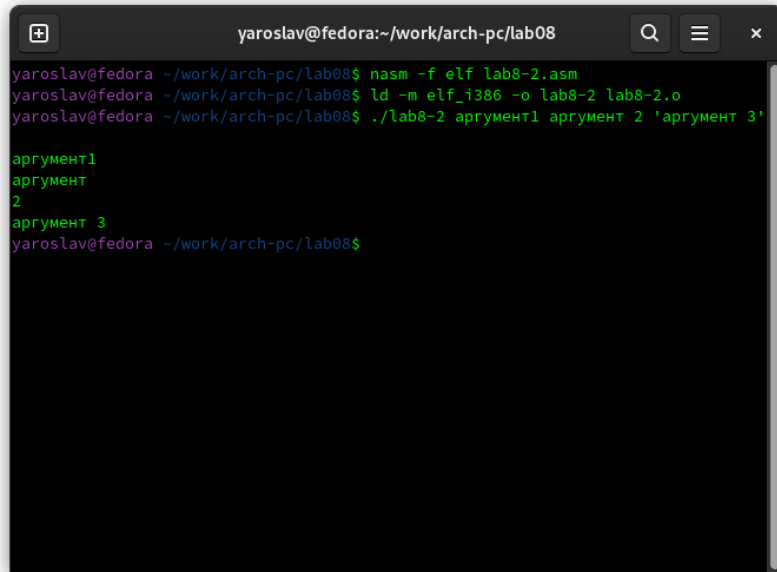
_start:
    pop ecx      ; Извлекаем из стека в 'ecx' количество
                  ; аргументов (первое значение в стеке)
    pop edx      ; Извлекаем из стека в 'edx' имя программы
                  ; (второе значение в стеке)
    sub ecx, 1    ; Уменьшаем 'ecx' на 1 (количество
                  ; 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. 2.8: Файл lab8-2.asm

8. Запускаем программу. Видим, что вывелось 4 аргумента (“аргумент 2” был считан как два аргумента).

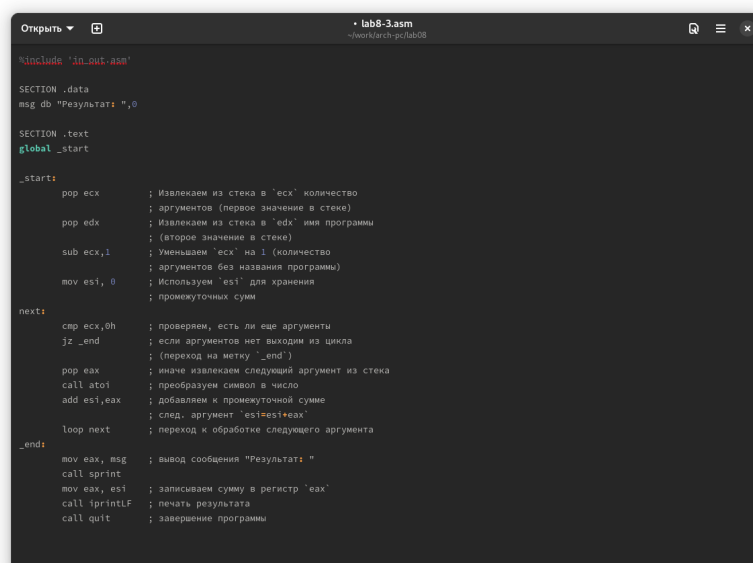


```
yaroslav@fedora: ~/work/arch-pc/lab08
yaroslav@fedora: ~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
yaroslav@fedora: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
yaroslav@fedora: ~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'

аргумент1
аргумент
2
аргумент 3
yaroslav@fedora: ~/work/arch-pc/lab08$
```

Рис. 2.9: Работа программы

9. Создаём файл lab8-3.asm и записываем туда листинг 8-3.



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

#include "in_out.asm"

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

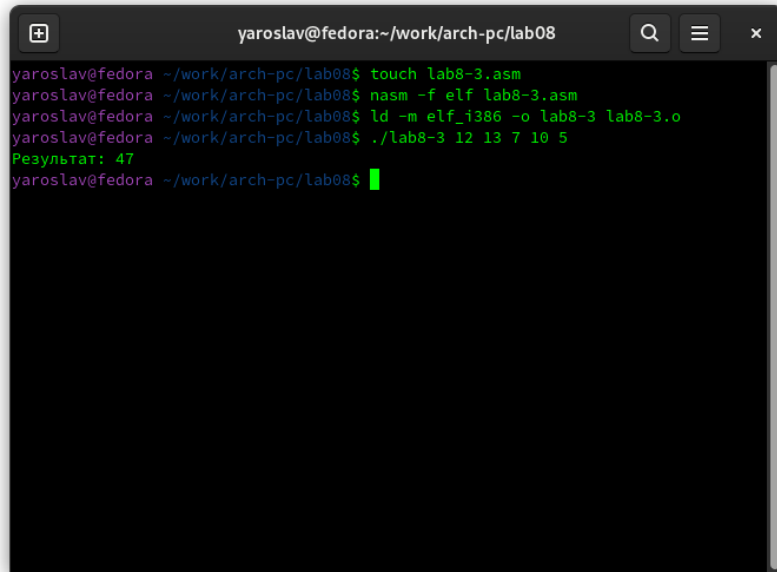
_start:
    pop ecx        ; Извлекаем из стека в 'ecx' количество
                  ; аргументов (первое значение в стеке)
    pop edx        ; Извлекаем из стека в 'edx' имя программы
                  ; (второе значение в стеке)
    sub ecx,1      ; Уменьшаем 'ecx' на 1 (количество
                  ; 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 sprintf
    mov eax,esi    ; записываем сумму в регистр 'eax'
    call iprintf   ; печать результата
    call quit      ; завершение программы
```

Рис. 2.10: Файл lab8-3.asm

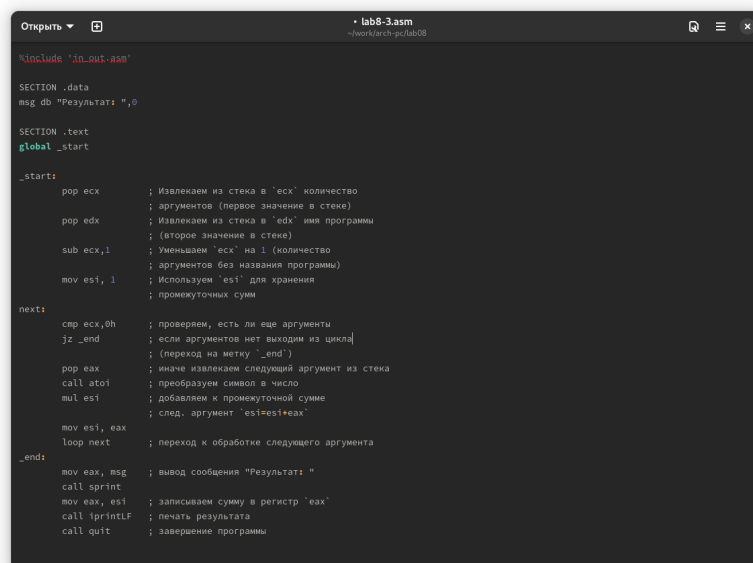
10. Проверяем работу.



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

Рис. 2.11: Работа программы

## 11. Меняем программу для вычисления произведения.



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

#include "in_out.asm"

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

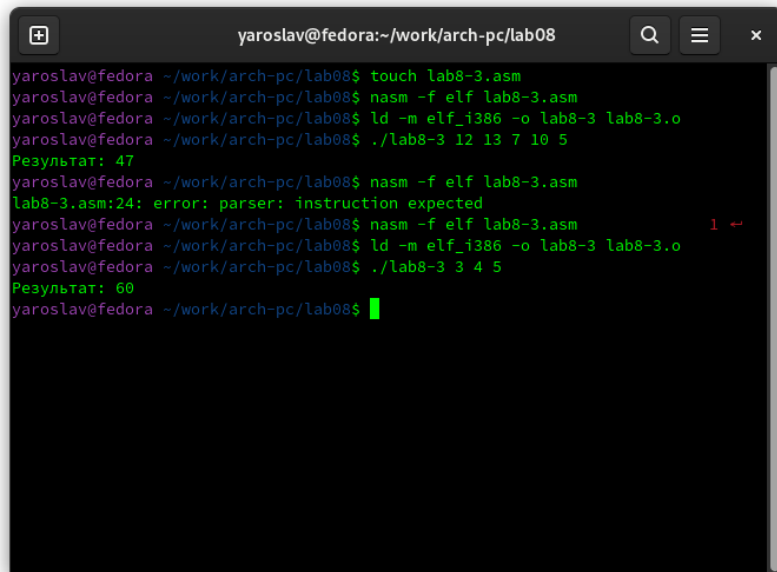
_start:
    pop ecx        ; Извлекаем из стека в 'ecx' количество
                  ; аргументов (первое значение в стеке)
    pop edx        ; Извлекаем из стека в 'edx' имя программы
                  ; (второе значение в стеке)
    sub ecx,1      ; Уменьшаем 'ecx' на 1 (количество
                  ; 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
    mul esi        ; добавляем к промежуточной сумме
                  ; след. аргумент 'esi=esi*eax'
    mov esi, eax   ; переход к обработке следующего аргумента
    loop next

_end:
    mov eax, msg   ; вывод сообщения "Результат: "
    call sprintf
    mov eax, esi   ; записываем сумму в регистр 'eax'
    call iprintf
    call quit      ; завершение программы
```

Рис. 2.12: Программа для вычисления произведения

## 12. Проверяем работу.

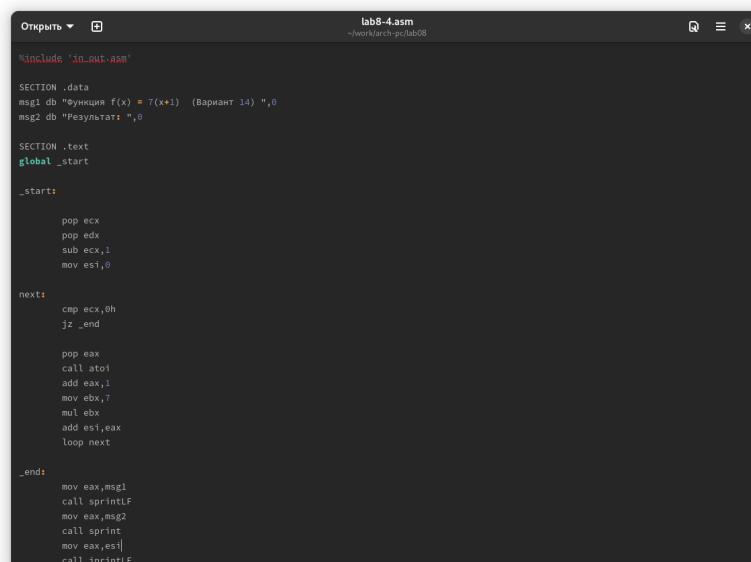


```
yaroslav@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
yaroslav@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
yaroslav@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
yaroslav@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
yaroslav@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
lab8-3.asm:24: error: parser: instruction expected
yaroslav@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
yaroslav@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
yaroslav@fedora:~/work/arch-pc/lab08$ ./lab8-3 3 4 5
Результат: 60
yaroslav@fedora:~/work/arch-pc/lab08$
```

Рис. 2.13: Работа программы

### 3 Выполнение самостоятельной работы

1. Создаём файл lab8-4.asm и пишем в нём текст программы (14 вариант).



```
Открыть ▾ lab8-4.asm
~/work/arch-pc/lab08

#include "in_out.asm"

SECTION .data
msg1 db "Функция f(x) = 7(x+1) (Вариант 14) ",0
msg2 db "Результат: ",0

SECTION .text
global _start

_start:

    pop ecx
    pop edx
    sub ecx,1
    mov esi,0

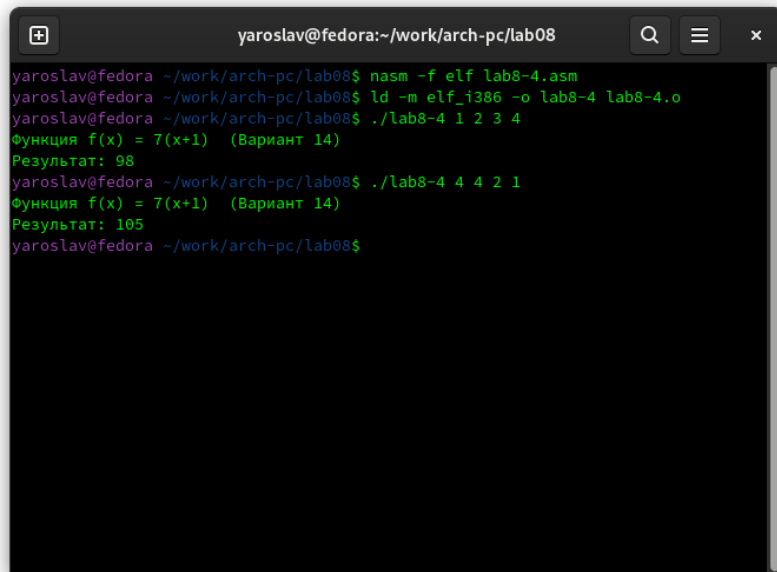
next:
    cmp ecx,0h
    jz _end

    pop eax
    call atoi
    add eax,1
    mov ebx,7
    mul ebx
    add esi,eax
    loop next

_end:
    mov eax,msg1
    call sprintf
    mov eax,msg2
    call sprintf
    mov eax,esi
    call sprintf
```

Рис. 3.1: Готовый lab7-3.asm

2. Проверяем работу.



```
yaroslav@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
yaroslav@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
yaroslav@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция  $f(x) = 7(x+1)$  (Вариант 14)
Результат: 98
yaroslav@fedora:~/work/arch-pc/lab08$ ./lab8-4 4 4 2 1
Функция  $f(x) = 7(x+1)$  (Вариант 14)
Результат: 105
yaroslav@fedora:~/work/arch-pc/lab08$
```

Рис. 3.2: Работа программы

## 4 Выводы

Были изучены циклы, были отработаны программы с использованием аргументов.