

# **Отчёт по лабораторной работе №8**

**Программирование цикла. Обработка аргументов командной строки**

Новикова Анастасия Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Реализация циклов в NASM . . . . .	7
3.2	Обработка аргументов командной строки . . . . .	9
3.3	Задание для самостоятельной работы . . . . .	12
<b>4</b>	<b>Выводы</b>	<b>15</b>

# Список иллюстраций

3.1	Создание каталога и файла в нём . . . . .	7
3.2	Редактирование программы . . . . .	7
3.3	Запуск программы . . . . .	8
3.4	Редактирование программы . . . . .	8
3.5	Запуск программы . . . . .	8
3.6	Редактирование программы . . . . .	9
3.7	Запуск программы . . . . .	9
3.8	Редактирование программы . . . . .	10
3.9	Запуск программы . . . . .	10
3.10	Редактирование программы . . . . .	10
3.11	Запуск программы . . . . .	11
3.12	Редактирование программы . . . . .	11
3.13	Запуск программы . . . . .	12
3.14	Создание файла . . . . .	12
3.15	Редактирование программы . . . . .	13
3.16	Запуск программы . . . . .	14
3.17	Повторный запуск программы . . . . .	14

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация циклов в NASM

Создаю директорию, в которой буду выполнять лабораторную работу, перехожу в созданный каталог и создаю файл lab8-1.asm, в котором буду выполнять первое задание (рис. 3.1).

```
aanovikoval23@fedora:~$ mkdir ~/work/arch-pc/lab08
aanovikoval23@fedora:~$ cd ~/work/arch-pc/lab08
aanovikoval23@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Рис. 3.1: Создание каталога и файла в нём

Ввожу текст кода из листинга 8.1 ((рис. 3.2). Эта программа запрашивает число N и выдает все числа перед N вместе с ним до 0 не включительно.



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

Рис. 3.2: Редактирование программы

Создаю исполняемый код (рис. 3.3). После его запуска убеждаюсь, что программа работает успешно.

```
aanovikova123@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aanovikova123@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 15
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
aanovikova123@fedora:~/work/arch-pc/lab08$
```

Рис. 3.3: Запуск программы

Теперь я редактирую код, добавив изменение значение регистра есх в цикле (рис. 3.4)

```
Открыть  *lab8-1.asm
~/work/arch-pc/lab08
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprintf
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call read
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
22 label:
23 sub ecx,1 ; 'ecx=ecx-1'
24 mov [N],ecx
25 mov eax,[N]
26 call sprintf
27 loop label ;
28 ; переход на 'label'
29 call quit
```

Рис. 3.4: Редактирование программы

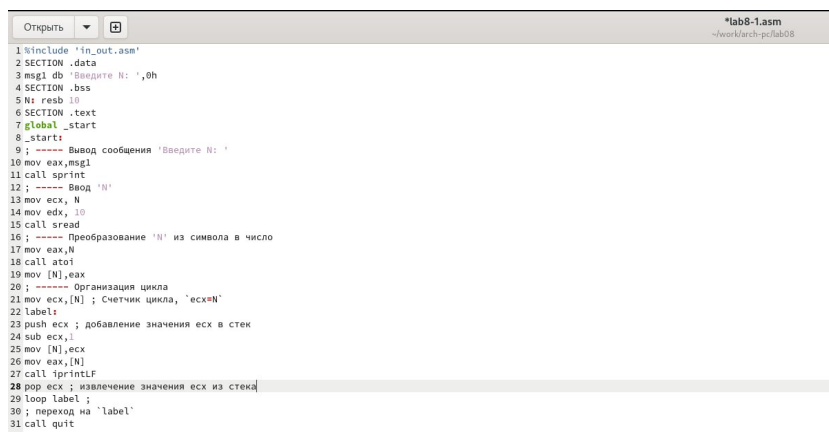
Запускаю программу. Теперь код передаёт значение через 2 числа. Регистр есх принимает значения 9, 7, 3, 5, 1. Число проходов по циклу не равно N. (рис. 3.5)

```
aanovikova123@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
```

Рис. 3.5: Запуск программы



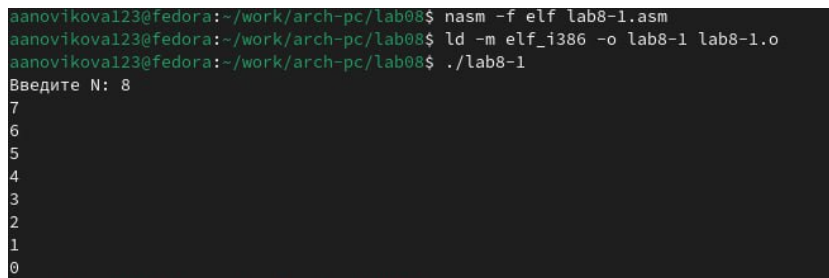
Еще раз редактирую код программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис. 3.6).



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

Рис. 3.6: Редактирование программы

Создаю и запускаю исполняемый файл (рис. 3.7). Теперь программа показывает все предыдущие числа до 0, не включая заданное N. Число проходов по циклу равно N.

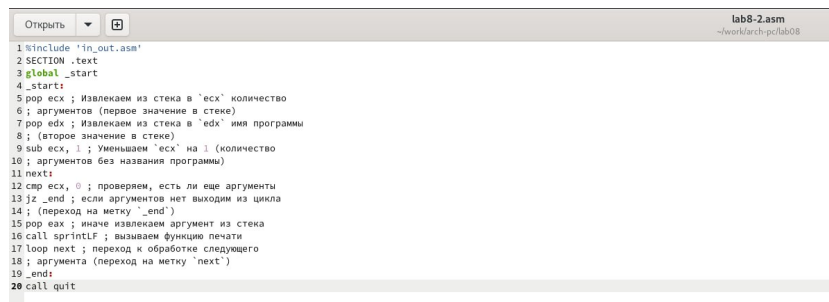


```
aanovikova123@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aanovikova123@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
6
5
4
3
2
1
0
```

Рис. 3.7: Запуск программы

## 3.2 Обработка аргументов командной строки

Создаю новый файл lab8-2.asm и ввожу в него код из листинга 8.2 (рис. 3.8). Данная программа позволяет выводить на экран аргументы командной строки.



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

Рис. 3.8: Редактирование программы

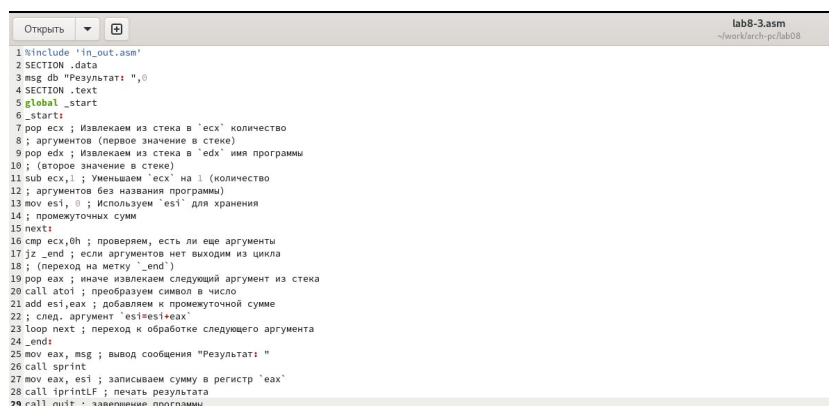
Запускаю исполняемый файл вместе с аргументами (аргумент1 аргумент 2 ‘аргумент 3’) (рис. 3.9). Программа обработала 4 аргумента: аргумент1 - 1-ый аргумент, аргумент - 2-ой аргумент, 2 - 3-ий аргумент, ‘аргумент 3’ - 4-ый аргумент.



```
aanovikova123@fedora: ~/work/arch-pc/lab08$ touch lab8-2.asm
aanovikova123@fedora: ~/work/arch-pc/lab08$ gedit lab8-2.asm
aanovikova123@fedora: ~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aanovikova123@fedora: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aanovikova123@fedora: ~/work/arch-pc/lab08$ ./lab8-2
aanovikova123@fedora: ~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 3.9: Запуск программы

Создаю новый файл lab8-3.asm, используя команду touch. Ввожу в него код из листинга 8.3 (рис. 3.10). Данная программа позволяет выводить на экран сумму аргументов командной строки.



```
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 sprintf
27 mov ecx, esi ; записываем сумму в регистр 'ecx'
28 call iprintf ; печать результата
29 call quit ; завершение программы
```

Рис. 3.10: Редактирование программы

Создаю и запускаю исполняемый файл вместе с аргументами (15, 5, 8, 11, 3) (рис. 3.11). Программа действительно выдаёт сумму всех аргументов.

```
aanovikova123@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ gedit lab8-3.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aanovikova123@fedora:~/work/arch-pc/lab08$ ./lab8-3 15 5 8 11 3
Результат: 42
aanovikova123@fedora:~/work/arch-pc/lab08$
```

Рис. 3.11: Запуск программы

Теперь редактирую код программы так, чтобы она выводила произведение всех аргументов (рис. 3.12).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx
8 pop edx
9 sub ecx,1
10 mov esi,1
11 next:
12 cmp ecx,0h
13 jz _end
14 pop eax
15 call atoi
16 mul esi
17 mov esi,eax
18 loop next
19 _end:
20 mov eax,msg
21 call sprintf
22 mov eax,esi
23 call iprintf
24 call quit
```

Рис. 3.12: Редактирование программы

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start
_start:

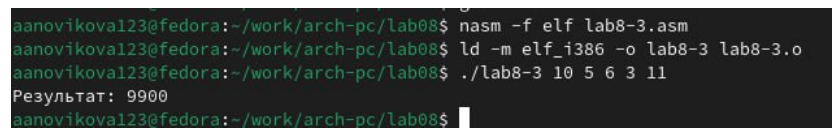
pop ecx
pop edx
sub ecx,1
mov esi, 1
next:
cmp ecx,0h
```

```

jz _end
pop eax
call atoi
mul esi
mov esi, eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit

```

Запускаю исполняемый файл вместе с аргументами (10, 5, 6, 3, 11) (рис. 3.13). Программа выдаёт произведение всех аргументов.



```

aanovikova123@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aanovikova123@fedora:~/work/arch-pc/lab08$ ./lab8-3 10 5 6 3 11
Результат: 9900
aanovikova123@fedora:~/work/arch-pc/lab08$

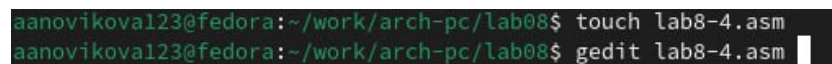
```

Рис. 3.13: Запуск программы

### 3.3 Задание для самостоятельной работы

#### **ВАРИАНТ 20**

Создаю файл lab8-4.asm в котором буду писать код для последней задачи (рис. 3.14)



```

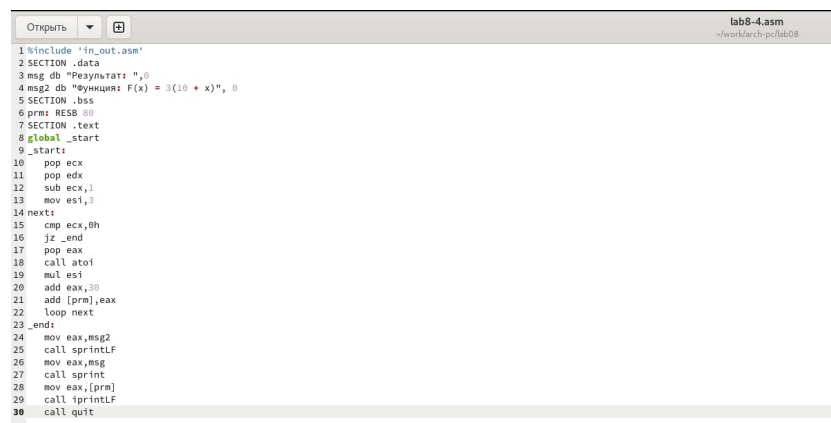
aanovikova123@fedora:~/work/arch-pc/lab08$ touch lab8-4.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ gedit lab8-4.asm

```

Рис. 3.14: Создание файла

Пишу код программы, который позволяет вывести сумму всех преобразованных аргументов. Преобразования я беру из варианта задания №20  $3(10 + x)$  (рис.

3.15)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 msg2 db "Функция: F(x) = 3(10 + x)", 0
5 SECTION .bss
6 prim RESB 80
7 SECTION .text
8 global _start
9 _start:
10 pop ecx
11 pop edx
12 sub ecx,1
13 mov esi,3
14 next:
15     cmp ecx,0h
16     jz _end
17     pop eax
18     call atoi
19     mul esi
20     add eax,30
21     add [prim],eax
22     loop next
23 _end:
24     mov eax,msg2
25     call sprintf
26     mov eax,msg
27     call sprintf
28     mov eax,[prim]
29     call sprintf
30     call quit
```

Рис. 3.15: Редактирование программы

Листинг написанной программы:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0
msg2 db "Функция: F(x) = 3(10 + x)", 0

SECTION .bss
prim: RESB 80

SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi,3
next:
    cmp ecx,0h
    jz _end
    pop eax
```

```

    call atoi
    mul esi
    add eax,30
    add [prm],eax
    loop next
_end:
    mov eax,msg2
    call sprintf
    mov eax,msg
    call sprintf
    mov eax,[prm]
    call sprintf
    call quit

```

Запускаю исполняемый файл вместе с аргументами (1, 2, 4, 5) (рис. 3.16). Программа выдаёт верную сумму всех преобразованных аргументов.

```

aanovikova123@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aanovikova123@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aanovikova123@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 4 5
Функция: F(x) = 3(10 + x)
Результат: 156

```

Рис. 3.16: Запуск программы

Повторно запускаю программу с другими аргументами (2, 5, 10, 11), чтобы убедиться, что всё работает верно (рис. 3.17). Программа выдает верный ответ.

```

aanovikova123@fedora:~/work/arch-pc/lab08$ ./lab8-4 2 5 10 11
Функция: F(x) = 3(10 + x)
Результат: 204
aanovikova123@fedora:~/work/arch-pc/lab08$ █

```

Рис. 3.17: Повторный запуск программы

## **4 Выводы**

В ходе лабораторной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.