

# **Отчёта по лабораторной работе №8**

**Программирование цикла. Обработка аргументов командной строки.**

Агиар Лионел Домингуш

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Реализация циклов в NASM . . . . .	6
3.2	Обработка аргументов командной строки. . . . .	8
3.3	Задание для самостоятельной работы . . . . .	11
<b>4</b>	<b>Выводы</b>	<b>13</b>

## Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code> . . . . .	6
3.2	Заполняем файл . . . . .	6
3.3	Запускаем файл и проверяем его работу . . . . .	6
3.4	Изменяем файл . . . . .	7
3.5	Запускаем файл и смотрим на его работу . . . . .	7
3.6	Редактируем файл . . . . .	8
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	8
3.8	Создаем файл командой <code>touch</code> . . . . .	8
3.9	Заполняем файл . . . . .	9
3.10	Смотрим на работу программ . . . . .	9
3.11	Создаем файл командой <code>touch</code> . . . . .	9
3.12	Заполняем файл . . . . .	10
3.13	Смотрим на работу программы . . . . .	10
3.14	Изменяем файл . . . . .	10
3.15	Проверяем работу файла(работает правильно) . . . . .	11
3.16	Создаем файл командой <code>touch</code> . . . . .	11
3.17	Пишем программу . . . . .	12
3.18	Смотрим на работу программы при $x_1=5$ $x_2=3$ $x_1=4$ (всё верно) . .	12
3.19	Смотрим на работу программы при $x_1=1$ $x_2=3$ $x_1=7$ (всё верно) . .	12

# **1 Цель работы**

Изучить работу циклов и обработкой аргументов командной строки.

## 2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

## 3 Выполнение лабораторной работы

### 3.1 Реализация циклов в NASM

Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. fig. 3.1).

Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. fig. 3.2).

Заполняем файл

Рис. 3.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.3).

```
lionelaguia@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lionelaguia@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
lionelaguia@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
Falta de segmentação (núcleo despejado)
lionelaguia@fedora:~/work/arch-pc/lab08$
```

Рис. 3.3: Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. fig. 3.4).

```
_start:
mov  eax,msg1
call sprint
mov  ecx, N
mov  edx, 10
call sread
mov  eax,N
call atoi
mov  [N],eax
mov  ecx,[N]
label:
sub  ecx,1.
mov  [N],ecx
mov  eax,[N]
call iprintLF
loop label
```

Рис. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5).

```
lionelaguilar@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lionelaguilar@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
lionelaguilar@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
Falta de segmentação (núcleo despejado)
lionelaguilar@fedora:~/work/arch-pc/lab08$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Регистр ecx принимает значения 9,7,5,3,1(на вход подается число 10, в цикле label данный регистр уменьшается на 2 командой sub и loop).

Число проходов цикла не соответствует числу N, так как уменьшается на 2.

Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис. fig. 3.6).

```

call atoi
mov [N],eax
mov ecx,[N]
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label

```

Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.7).

```

lionelaguia@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lionelaguia@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
lionelaguia@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
Falta de segmentação (núcleo despejado)
lionelaguia@fedora:~/work/arch-pc/lab08$

```

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

В данном случае число проходов цикла равна числу N.

## 3.2 Обработка аргументов командной строки.

Создаем новый файл (рис. fig. 3.8).

```

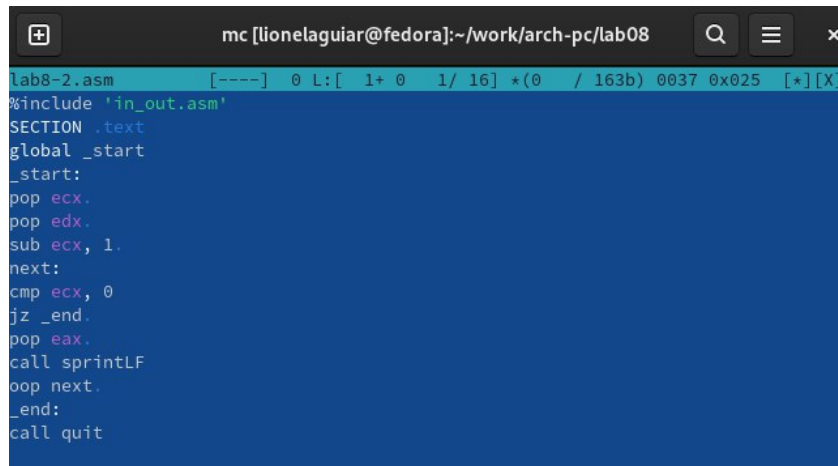
lionelaguia@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
lionelaguia@fedora:~/work/arch-pc/lab08$

```

Рис. 3.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. fig. 3.9).

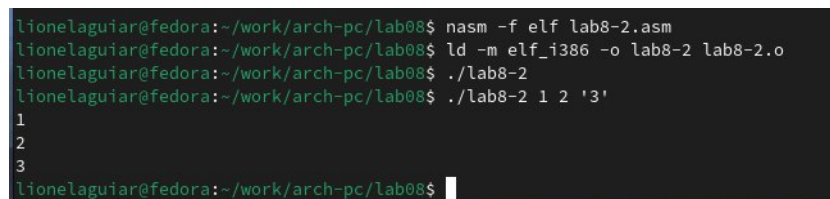




```
lab8-2.asm [----] 0 L: [ 1+ 0 1/ 16] *(0 / 163b) 0037 0x025 [*][X]
#include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx.
pop edx.
sub ecx, 1.
next:
cmp ecx, 0
jz _end.
pop eax.
call sprintf
oop next.
_end:
call quit
```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. fig. 3.10).

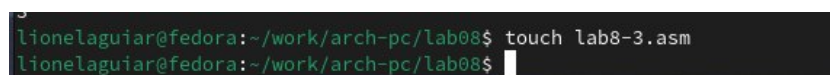


```
lionelaguair@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
lionelaguair@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
lionelaguair@fedora:~/work/arch-pc/lab08$ ./lab8-2
1
2
3
lionelaguair@fedora:~/work/arch-pc/lab08$
```

Рис. 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

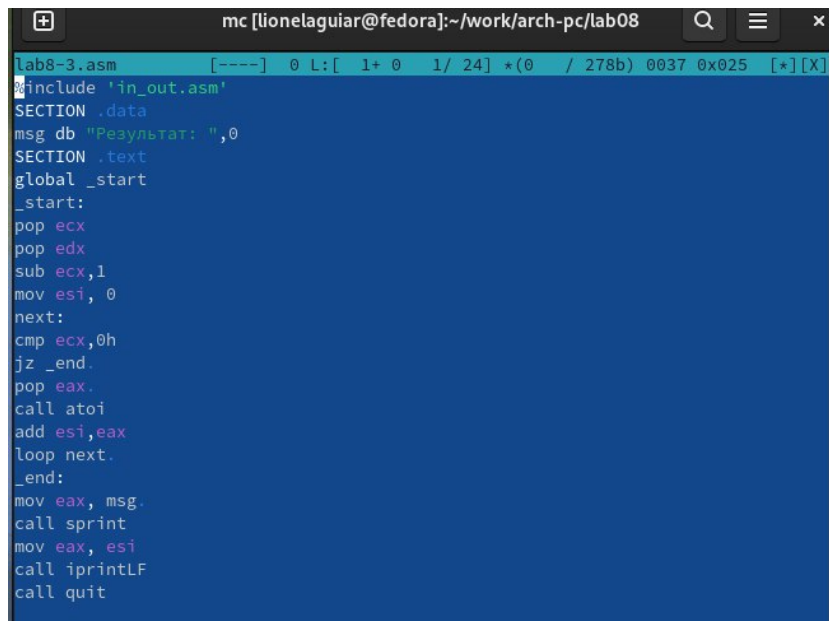
Создаем новый файл lab8-3.asm (рис. fig. 3.11).



```
lionelaguair@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
lionelaguair@fedora:~/work/arch-pc/lab08$
```

Рис. 3.11: Создаем файл командой touch

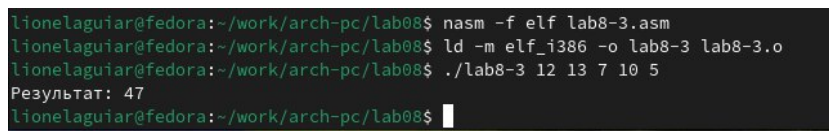
Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. fig. 3.12).

A screenshot of a text editor window titled 'mc [lionelaguair@fedora]:~/work/arch-pc/lab08'. The editor shows the assembly file 'lab8-3.asm'. The code includes a data section with a message 'Результат: ', a text section with a loop that reads input and calculates a sum, and a final print statement. The code is as follows:

```
lab8-3.asm
[----] 0 L: [ 1+ 0 1/ 24] *(0 / 278b) 0037 0x025 [*][X]
%include 'in_out.asm'
SECTION .data
msg 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 esi,eax
loop next.
_end:
mov eax, msg.
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 3.12: Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.13).

A screenshot of a terminal window showing the compilation and execution of the assembly program. The commands and output are as follows:

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

Рис. 3.13: Смотрим на работу программы

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. fig. 3.14).

A screenshot of the text editor showing the assembly code being modified. The code is as follows:

```
next:
cmp ecx,0h
jz _end.
pop eax.
call atoi
mul esi
mov esi,eax
loop next.
_end:
```

Рис. 3.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.15).

```

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

```

Рис. 3.15: Проверяем работу файла(работает правильно)

### 3.3 Задание для самостоятельной работы

#### ВАРИАНТ-20

1. Напишите программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x_i$  передаются как аргументы. Вид функции  $f(x)$  выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах  $x = x_1, x_2, \dots, x_n$ .

Создаем новый файл (рис. fig. 3.16).

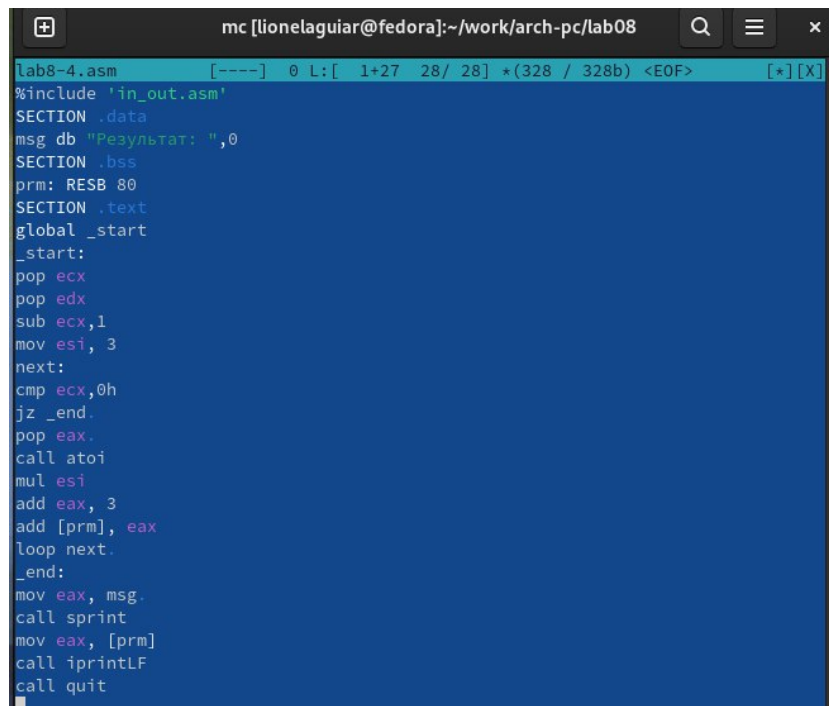
```

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

```

Рис. 3.16: Создаем файл командой touch

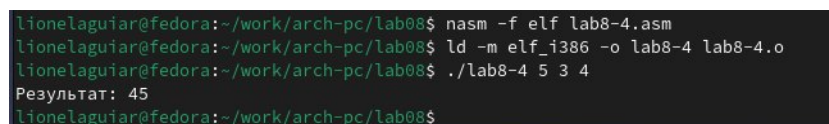
Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения  $3(10+x)$  (рис. fig. 3.17).

A screenshot of a code editor window titled 'mc [lionelaguair@fedora]:~/work/arch-pc/lab08'. The editor shows the assembly file 'lab8-4.asm' with the following code:

```
lab8-4.asm      [----]  0 L: [ 1+27  28/ 28] *(328 / 328b) <EOF>  [*][X]
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .bss
prm: 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, 3
add [prm], eax
loop next.
_end:
mov eax, msg.
call sprint
mov eax, [prm]
call iprintLF
call quit
```

Рис. 3.17: Пишем программу

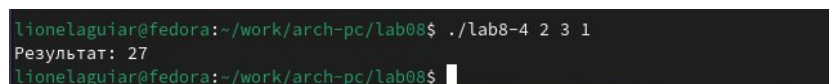
Транслируем файл и смотрим на работу программы (рис. fig. 3.18).

A screenshot of a terminal window showing the following commands and output:

```
lionelaguair@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
lionelaguair@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
lionelaguair@fedora:~/work/arch-pc/lab08$ ./lab8-4 5 3 4
Результат: 45
lionelaguair@fedora:~/work/arch-pc/lab08$
```

Рис. 3.18: Смотрим на работу программы при  $x_1=5$   $x_2=3$   $x_3=4$ (всё верно)

Транслируем файл и смотрим на работу программы (рис. fig. 3.19).

A screenshot of a terminal window showing the following commands and output:

```
lionelaguair@fedora:~/work/arch-pc/lab08$ ./lab8-4 2 3 1
Результат: 27
lionelaguair@fedora:~/work/arch-pc/lab08$
```

Рис. 3.19: Смотрим на работу программы при  $x_1=1$   $x_2=3$   $x_3=7$ (всё верно)

## 4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.