

# **Отчёта по лабораторной работе №8**

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

Жозе Рамос 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 . . . . .	7
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) . . . . .	10
3.16	Создаем файл командой <code>touch</code> . . . . .	11
3.17	Пишем программу . . . . .	11
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

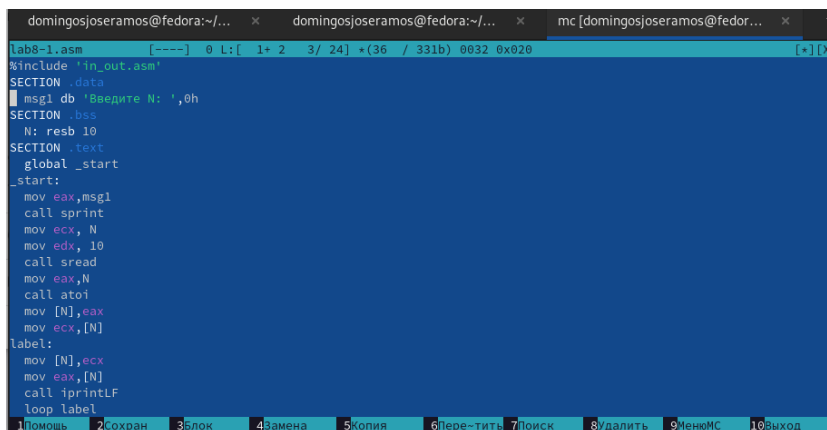
### 3.1 Реализация циклов в NASM

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

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

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).



```
lab8-1.asm  [----]  0 L: [ 1+ 2 3/ 24] *(36 / 331b) 0032 0x020  [*)(X)]
%include 'in_out.asm'
SECTION .data
    msg1 db 'Введите N: ',0h
SECTION .bss
    N: resb 10
SECTION .text
    global _start
_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:
    mov [N],ecx
    mov eax,[N]
    call iprintLF
    loop label
```

Рис. 3.2: Заполняем файл

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

```

domingosjoseramos@fedora:~/work/arch-pc/lab8$ nasm -f elf lab8-1.asm
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
domingosjoseramos@fedora:~/work/arch-pc/lab8$

```

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 spread
mov eax,N
call atoi
mov [N],eax
mov ecx,[N]
label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label

```

Рис. 3.4: Изменяем файл

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

```

domingosjoseramos@fedora:~/work/arch-pc/lab8$ nasm -f elf lab8-1.asm
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ./lab8-1
Введите N: 10
9
7
5
3
1
Ошибка сегментирования (образ памяти сброшен на диск)
domingosjoseramos@fedora:~/work/arch-pc/lab8$

```

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).

```

domingosjoseramos@fedora: ~/work/arch-pc/lab8$ nasm -f elf lab8-1.asm
domingosjoseramos@fedora: ~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o
domingosjoseramos@fedora: ~/work/arch-pc/lab8$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
Ошибка сегментирования (образ памяти сброшен на диск)
domingosjoseramos@fedora: ~/work/arch-pc/lab8$

```

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).

```

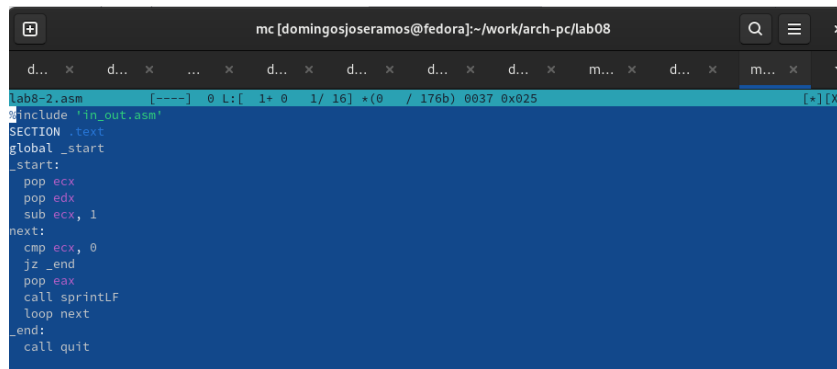
domingosjoseramos@fedora: ~/work/arch-pc/lab8$ touch lab8-2.asm
domingosjoseramos@fedora: ~/work/arch-pc/lab8$

```

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).

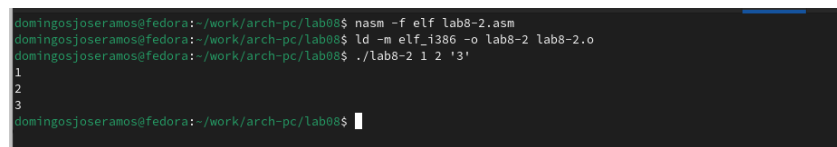




```
lab8-2.asm [---] 0 L: [ 1+ 0 1/ 16] *(0 / 176b) 0037 0x025 [*][X]
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx, 1
next:
    cmp ecx, 0
    jz _end
    pop eax
    call sprintf
    loop next
_end:
    call quit
```

Рис. 3.9: Заполняем файл

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

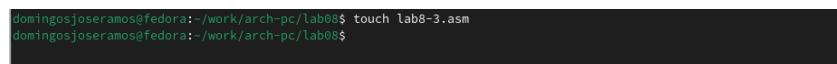


```
domingosjoseramos@fedora: ~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
domingosjoseramos@fedora: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
domingosjoseramos@fedora: ~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
domingosjoseramos@fedora: ~/work/arch-pc/lab08$
```

Рис. 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

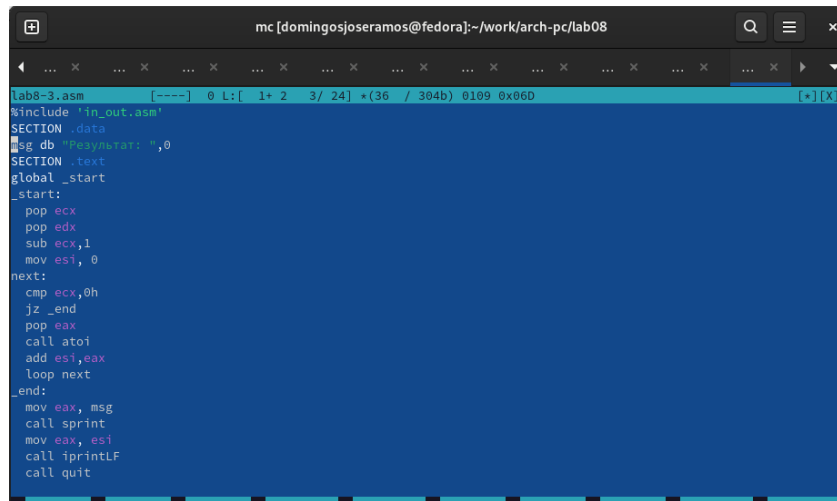
Создаем новый файл lab8-3.asm (рис. fig. 3.11).



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

Рис. 3.11: Создаем файл командой touch

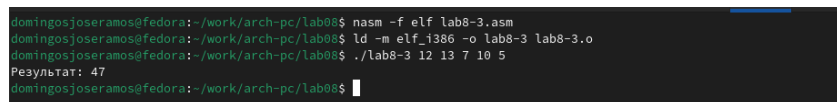
Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. fig. 3.12).



```
lab8-3.asm  [----]  0 L: [ 1+ 2 3/ 24] +(36 / 304b) 0109 0x06D [*)(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 iprintfLF
    call quit
```

Рис. 3.12: Заполняем файл

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



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

Рис. 3.13: Смотрим на работу программы

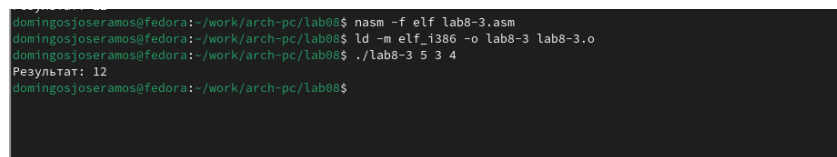
Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. fig. 3.14).



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

Рис. 3.14: Изменяем файл

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



```
domingosjoseramos@fedora: ~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
domingosjoseramos@fedora: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
domingosjoseramos@fedora: ~/work/arch-pc/lab08$ ./lab8-3 5 3 4
Результат: 12
domingosjoseramos@fedora: ~/work/arch-pc/lab08$
```

Рис. 3.15: Проверяем работу файла(работает правильно)

### 3.3 Задание для самостоятельной работы

#### ВАРИАНТ-13

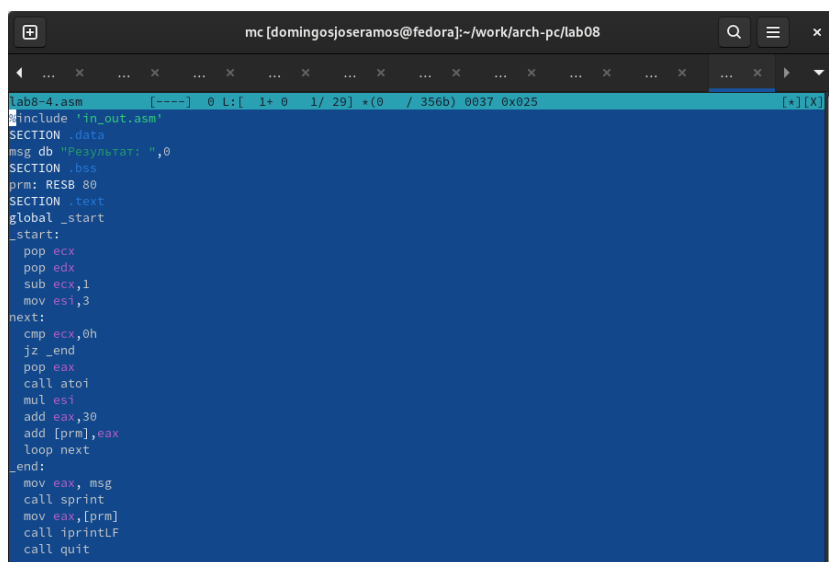
1. Напишите программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x_i$  передаются как аргументы. Вид функции  $f(x)$  выбрать из таблицы 8.1 вариантов заданий в соответствии 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).

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



```
lab8-4.asm 0 L: [ 1+ 0 1/ 29] * (0 / 356b) 0037 0x025
include "in_out.asm"
SECTION .data
msg db "Pexynat: ",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,30
    add [prm],eax
    loop next
_end:
    mov eax,msg
    call sprint
    mov eax,[prm]
    call iprintlnLF
    call quit
```

Рис. 3.17: Пишем программу

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

```
domingosjoseramos@fedora:~/work/arch-pc/lab8$ nasm -f elf lab8-4.asm
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-4 lab8-4.o
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ./lab8-4 5 3 4
Результат: 126
domingosjoseramos@fedora:~/work/arch-pc/lab8$
```

Рис. 3.18: Смотрим на работу программы при  $x1=5$   $x2=3$   $x1=4$ (всё верно)

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

```
domingosjoseramos@fedora:~/work/arch-pc/lab8$ nasm -f elf lab8-4.asm
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-4 lab8-4.o
domingosjoseramos@fedora:~/work/arch-pc/lab8$ ./lab8-4 1 3 7
Результат: 123
domingosjoseramos@fedora:~/work/arch-pc/lab8$
```

Рис. 3.19: Смотрим на работу программы при  $x1=1$   $x2=3$   $x1=7$ (всё верно)

## 4 Выводы

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