

# **Отчёт по лабораторной работе №8**

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

Югай Александр Витальевич

# 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. . . . .	11
3.3	Задание для самостоятельной работы . . . . .	15
<b>4</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code> . . . . .	7
3.2	Заполняем файл . . . . .	8
3.3	Запускаем файл и проверяем его работу . . . . .	9
3.4	Изменяем файл . . . . .	9
3.5	Запускаем файл и смотрим на его работу . . . . .	9
3.6	Редактируем файл . . . . .	10
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	10
3.8	Создаем файл командой <code>touch</code> . . . . .	11
3.9	Заполняем файл . . . . .	12
3.10	Смотрим на работу программ . . . . .	12
3.11	Создаем файл командой <code>touch</code> . . . . .	13
3.12	Заполняем файл . . . . .	14
3.13	Смотрим на работу программы . . . . .	15
3.14	Изменяем файл . . . . .	15
3.15	Проверяем работу файла . . . . .	15
3.16	Создаем файл командой <code>touch</code> . . . . .	16
3.17	Пишем программу . . . . .	17
3.18	Смотрим на работу программы при $x_1=2$ $x_2=5$ $x_3=3$ . . . . .	18
3.19	Смотрим на работу программы при $x_1=3$ $x_2=1$ $x_3=4$ . . . . .	18

## Список таблиц

# 1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

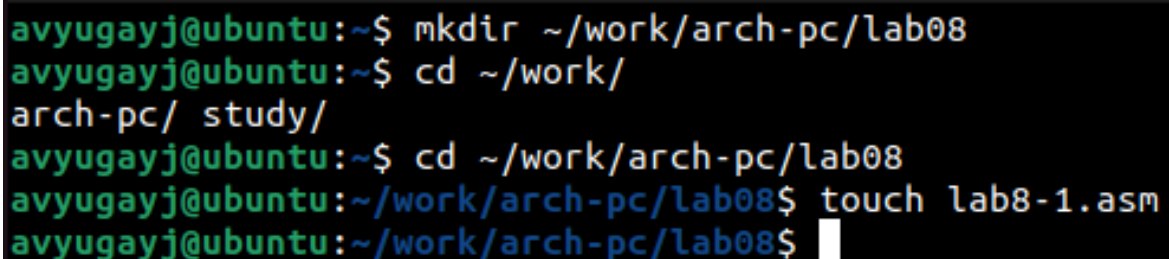
## 2 Задание

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



```
avyugayj@ubuntu:~$ mkdir ~/work/arch-pc/lab08
avyugayj@ubuntu:~$ cd ~/work/
arch-pc/ study/
avyugayj@ubuntu:~$ cd ~/work/arch-pc/lab08
avyugayj@ubuntu:~/work/arch-pc/lab08$ touch lab8-1.asm
avyugayj@ubuntu:~/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

```

GNU nano 6.2
;-----
; Программа вывода значений регистра 'ecx'
;-----
#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

```

Рис. 3.2: Заполняем файл



Создаем исполняемый файл и запускаем его

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

Рис. 3.3: Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле

```
label:
sub ecx,1sub ecx,1 ; `ecx=ecx-1`
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
```

Рис. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его

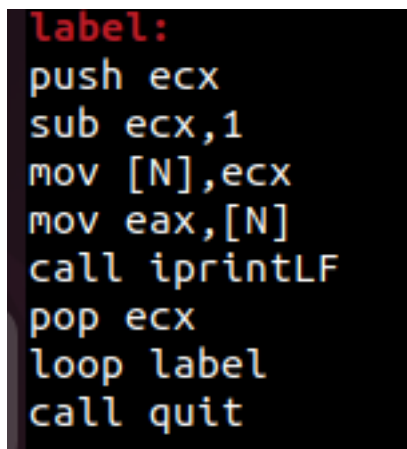
```
avyugayj@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avyugayj@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avyugayj@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
```

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.

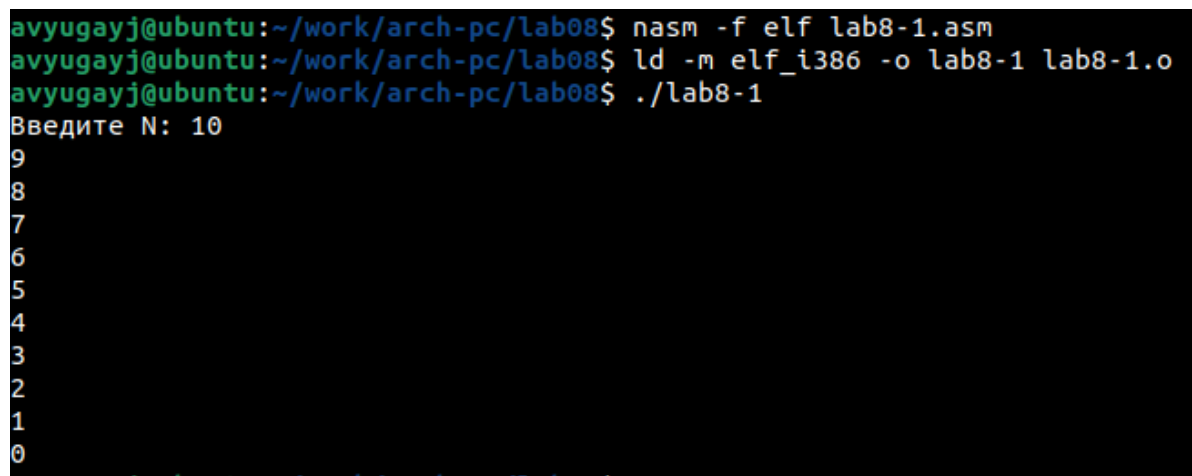
Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало



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

Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его



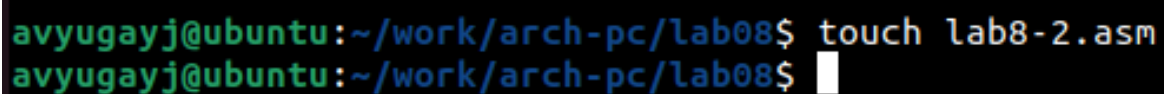
```
avyugayj@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avyugayj@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avyugayj@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

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

A terminal window with a black background and green text. The prompt is 'avyugayj@ubuntu:~/work/arch-pc/lab08\$'. The command 'touch lab8-2.asm' has been entered and executed. The next line shows the prompt 'avyugayj@ubuntu:~/work/arch-pc/lab08\$' followed by a white cursor block.

```
avyugayj@ubuntu:~/work/arch-pc/lab08$ touch lab8-2.asm  
avyugayj@ubuntu:~/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

```

;-----
; Обработка аргументов командной строки
;-----
%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. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы

```

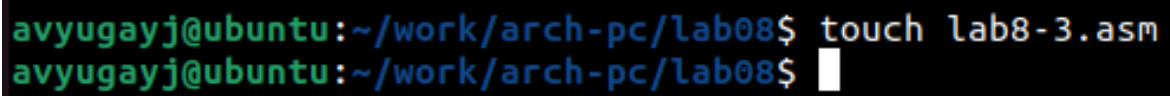
avyugayj@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
avyugayj@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
avyugayj@ubuntu:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3

```

Рис. 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm



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

Рис. 3.11: Создаем файл командой touch

Открываем файл и заполняем его в соответствии с листингом 8.3

```

#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 sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы

```

Рис. 3.12: Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы

```

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

```

Рис. 3.13: Смотрим на работу программы

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений

```

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
mov esi,eax; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента

```

Рис. 3.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы

```

avyugayj@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
avyugayj@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
avyugayj@ubuntu:~/work/arch-pc/lab08$ ./lab8-3 5 5 4
Результат: 100
avyugayj@ubuntu:~/work/arch-pc/lab08$ █

```

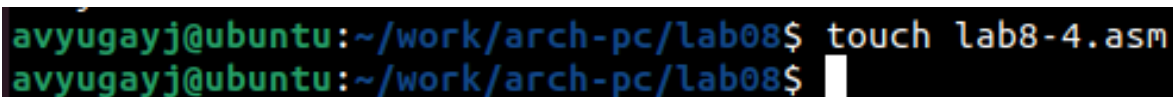
Рис. 3.15: Проверяем работу файла

### 3.3 Задание для самостоятельной работы

Вариант 3

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



```
avyugayj@ubuntu:~/work/arch-pc/lab08$ touch lab8-4.asm  
avyugayj@ubuntu:~/work/arch-pc/lab08$
```

Рис. 3.16: Создаем файл командой touch

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения  $10x-5$



```

#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .bss
ans: RESB 80
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,10
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul esi
sub eax,5
add [ans],eax
loop next
_end:
mov eax, msg
call sprint
mov eax,[ans]
call iprintLF
call quit

```

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы

```
avyugayj@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
avyugayj@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
avyugayj@ubuntu:~/work/arch-pc/lab08$ ./lab8-4 2 5 3
Результат: 85
```

Рис. 3.18: Смотрим на работу программы при  $x_1=2$   $x_2=5$   $x_3=3$

Транслируем файл и смотрим на работу программы

```
avyugayj@ubuntu:~/work/arch-pc/lab08$ ./lab8-4 3 1 4
Результат: 65
avyugayj@ubuntu:~/work/arch-pc/lab08$
```

Рис. 3.19: Смотрим на работу программы при  $x_1=3$   $x_2=1$   $x_3=4$

## 4 Выводы

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