

Отчёта по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Гурбанов Сарча

Содержание

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

Список иллюстраций

1.	<u>Создаем каталог с помощью команды mkdir и файл с помощью команды touch</u>	6
2.	<u>Заполняем файл</u>	7
3.	<u>Запускаем файл и проверяем его работу</u>	7
4.	<u>Изменяем файл</u>	8
5.	<u>Запускаем файл и смотрим на его работу</u>	8
6.	<u>Редактируем файл</u>	9
7.	<u>Проверяем, сошелся ли наш вывод с данным в условии выводом</u>	9
8.	<u>Создаем файл командой touch</u>	9
9.	<u>Заполняем файл</u>	10
10.	<u>Смотрим на работу программ</u>	10
11.	<u>Создаем файл командой touch</u>	10
12.	<u>Заполняем файл</u>	11
13.	<u>Смотрим на работу программы</u>	11
14.	<u>Изменяем файл</u>	12
15.	<u>Проверяем работу файла(работает правильно)</u>	12
16.	<u>Создаем файл командой touch</u>	13
17.	<u>Пишем программу</u>	13
18.	<u>Смотрим на работу программы при x1=5 x2=3 x1=4(всё верно)</u>	13
19.	<u>Смотрим на работу программы при x1=1 x2=3 x1=7(всё верно)</u>	14

1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

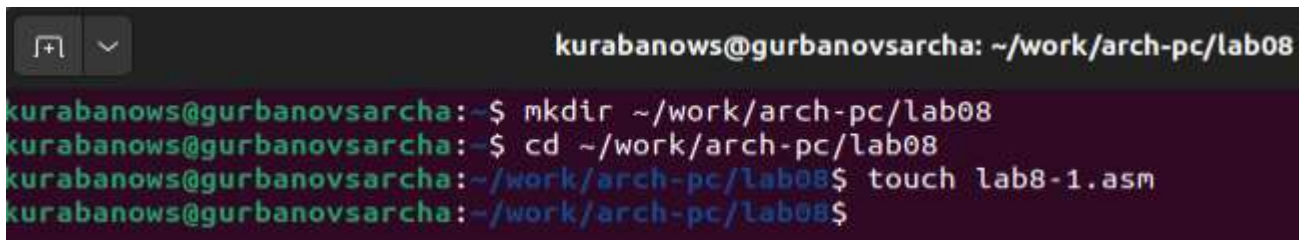
2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

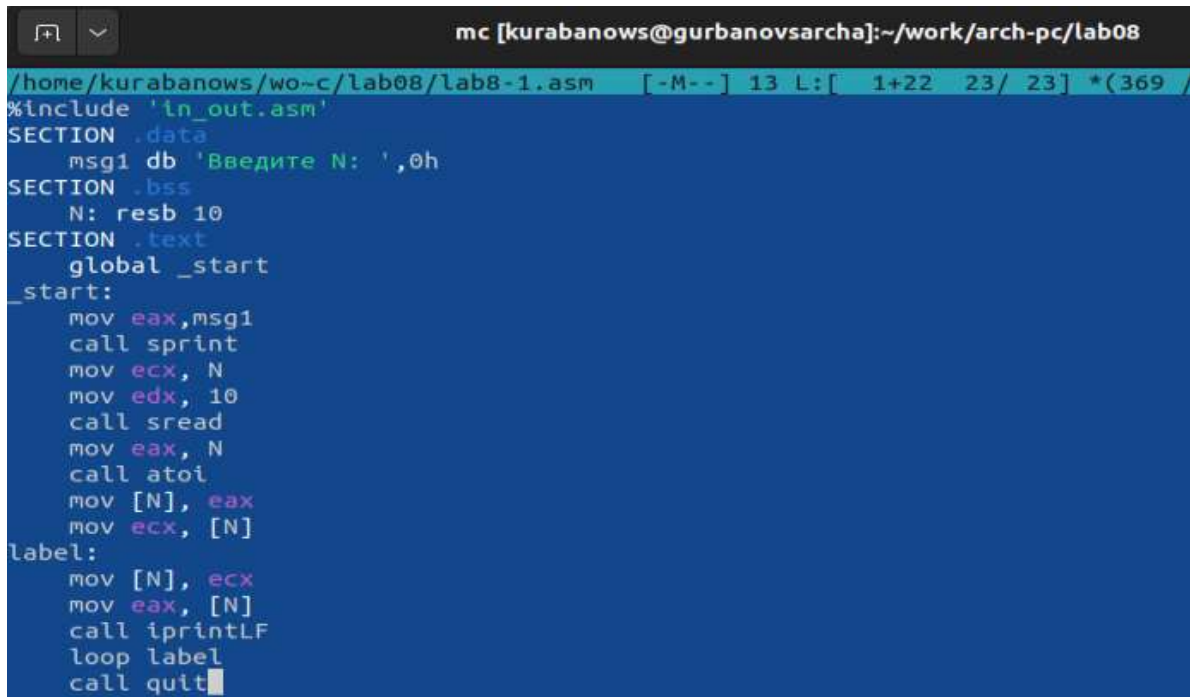
Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. [3.1](#)).

A terminal window with a dark background. The prompt is 'kurabanows@gurbanovsarcha: ~/work/arch-pc/lab08'. The user enters 'mkdir ~/work/arch-pc/lab08', then 'cd ~/work/arch-pc/lab08', and finally 'touch lab8-1.asm'. The prompt changes to 'kurabanows@gurbanovsarcha: ~/work/arch-pc/lab08\$' after each command.

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

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

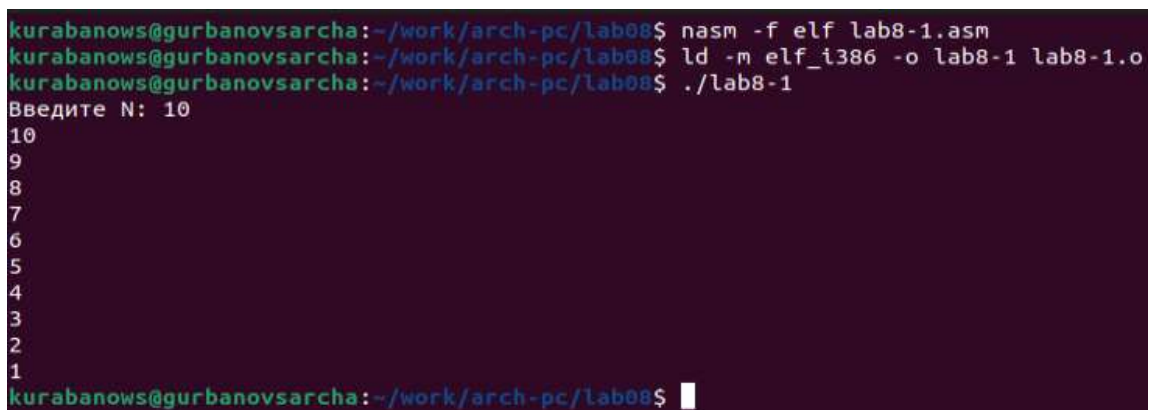
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. [3.2](#)).

A screenshot of a text editor window titled 'mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab08'. The editor shows the contents of the file '/home/kurabanows/work/arch-pc/lab08/lab8-1.asm'. The code is written in assembly language and includes sections for data, bss, and text. It defines a message 'Введите N: ', a buffer 'N' of size 10, and a loop that reads input, converts it to an integer, and prints it. The code ends with a 'quit' instruction.

```
mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab08
/home/kurabanows/work/arch-pc/lab08/lab8-1.asm [-M--] 13 L: [ 1+22 23/ 23] *(369 /
%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
    call quit
```

Рис. 3.2: Заполняем файл

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

A screenshot of a terminal window showing the compilation and execution of the assembly file. The user runs 'nasm -f elf lab8-1.asm' to create an object file, then 'ld -m elf_i386 -o lab8-1 lab8-1.o' to create an executable. Finally, they run './lab8-1', which prompts 'Введите N: 10' and displays a vertical list of numbers from 10 down to 1.

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
kurabanows@gurbanovsarcha:~/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. 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. 3.5).

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
Segmentation fault (core dumped)
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ s

```

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. 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. 3.7).

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
Segmentation fault (core dumped)
kurabanows@gurbanovsarcha:~/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. 3.8).

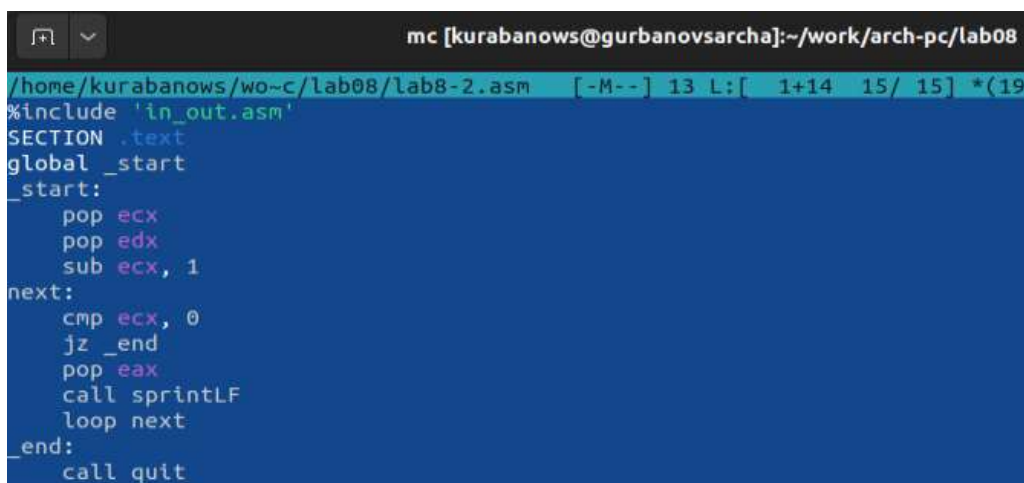
```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ touch lab8-2.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$

```

Рис. 3.8: Создаем файл командой touch

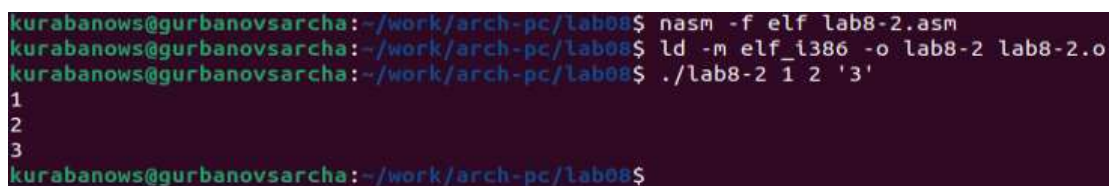
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. 3.9).

A screenshot of the Midnight Commander file manager. The title bar shows 'mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab08'. The main window displays the file 'lab8-2.asm' with the following assembly code:

```
%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. 3.10).

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

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

Рис. 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

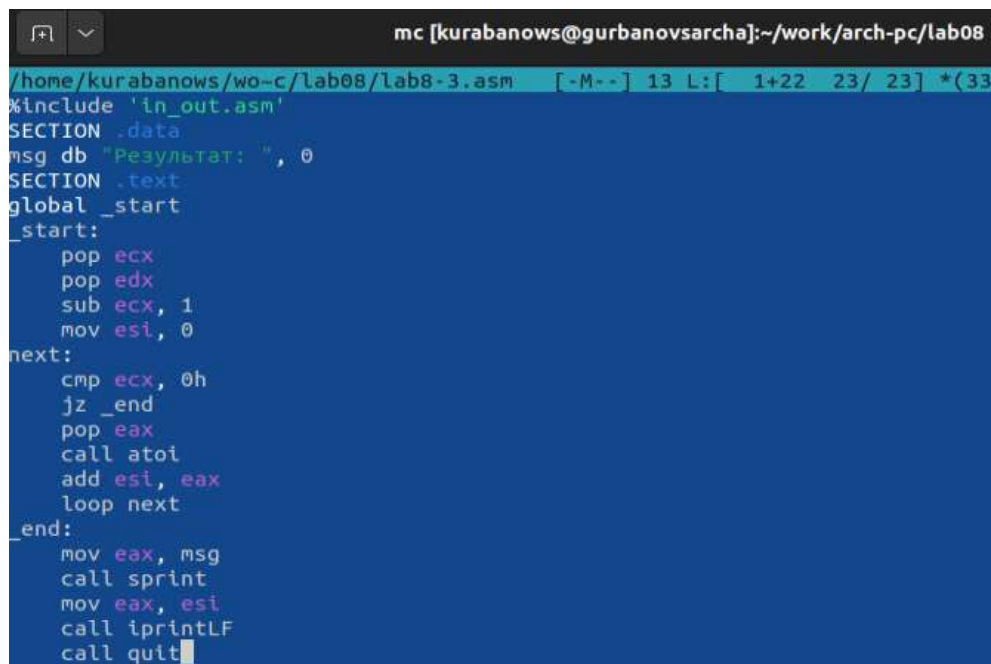
Создаем новый файл lab8-3.asm (рис. 3.11).

A screenshot of a terminal window showing the command to create a new file:

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

Рис. 3.11: Создаем файл командой touch

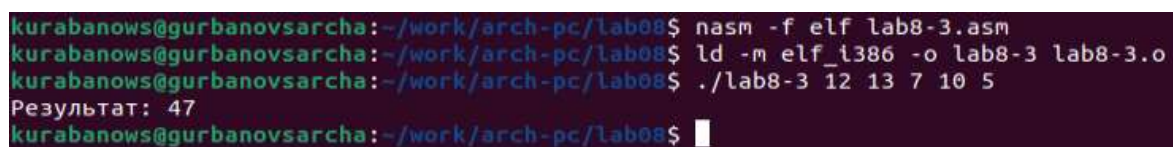
Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. 3.12).



```
mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab08
/home/kurabanows/work/arch-pc/lab08/lab8-3.asm [-M--] 13 L: [ 1+22 23/ 23] *(33
%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. 3.13).



```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
kurabanows@gurbanovsarcha:~/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. 3.14).

```

    mov esi, 1
next:
    cmp ecx, 0h
    jz _end
    pop eax
    call atoi
    imul esi, eax
    loop next
_end:

```

Рис. 3.14: Изменяем файл

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

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ./lab8-3 5 3 4
Результат: 60
kurabanows@gurbanovsarcha:~/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 $x = 1, 2, \dots, n$, т.е. программа должна выводить значение $f(1) + f(2) + \dots + f(n)$. Значения n 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 $n = 1, 2, \dots, 10$.

Создаем новый файл (рис. 3.16).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ touch lab8-4.asm
kurabanows@gurbanovsarcha:~/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. 3.17).

```
mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab08
/home/ku-b8-4.asm [-M--] 13 L:[ 1+26 27/ 27] *(393 / 3
%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, 30
    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. 3.18).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ./lab8-4 5 3 4
Результат: 126
kurabanows@gurbanovsarcha:~/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. 3.19).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab08$ ./lab8-4 1 3 7
Результат: 123
kurabanows@gurbanovsarcha:~/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.