

# **Лабораторная работа №8**

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

Демин Даниил

# Содержание

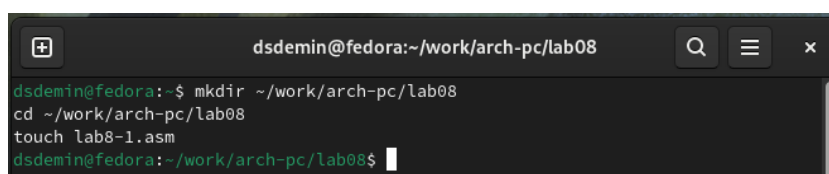
<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>4</b>
<b>3</b>	<b>Выполнение самостоятельной работы</b>	<b>15</b>
<b>4</b>	<b>Выводы</b>	<b>18</b>

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

## 2 Выполнение лабораторной работы

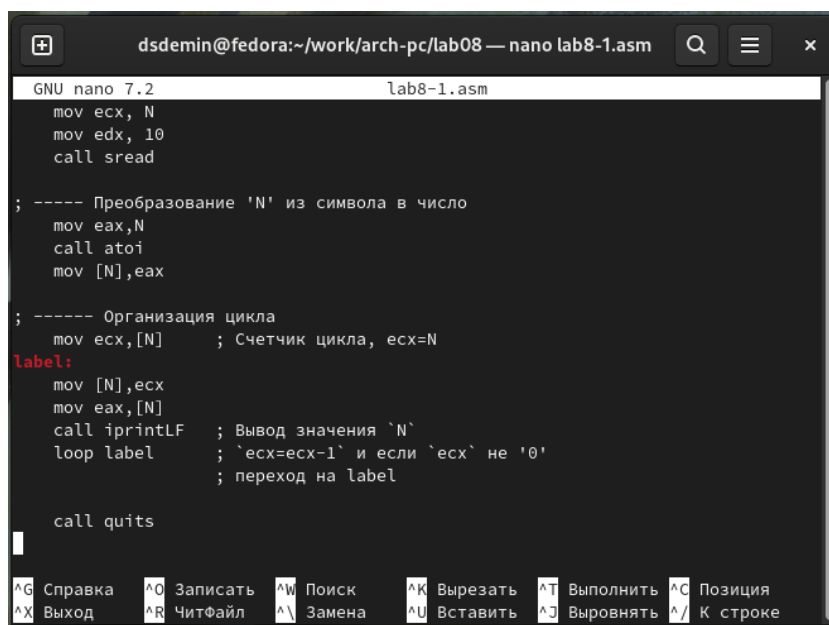
Создал и перешел в директорию для лабораторной работы. Создал файл lab7-8.asm (рис. 2.1).



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

Рис. 2.1: Папка для лабораторной работы

Переписал код с лабараторной работы(рис. 2.2).



```
GNU nano 7.2 lab8-1.asm
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 quits

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/_ К строке
```

Рис. 2.2: Листинг кода

Листинг кода:

```
;-----  
; 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

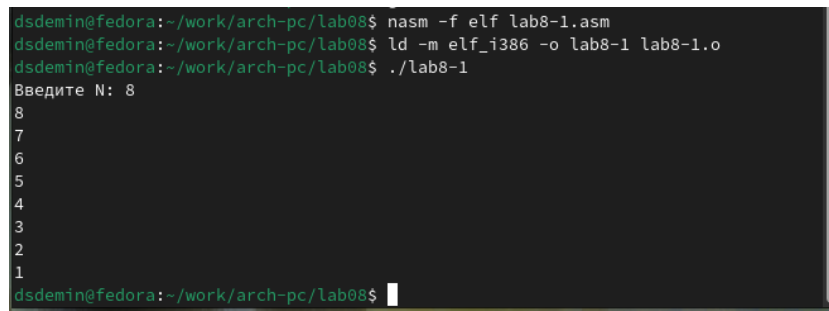
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`

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

```

Создала и запустил исполняемый файл. (рис. 2.3).



```

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

```

Рис. 2.3: Результат выполнения

Заменяю часть кода на другой из лабораторной работы.

Листинг кода:

```

; 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:
    sub ecx,1
    mov [N],ecx

```

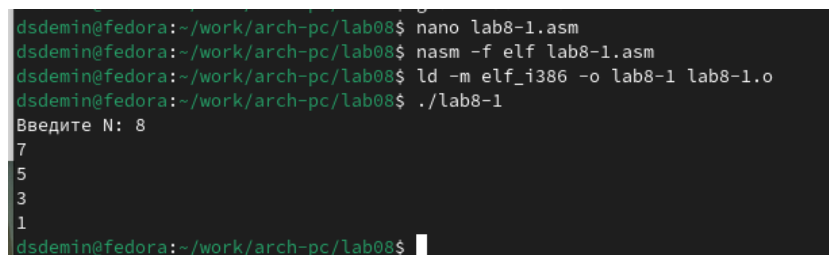
```

mov eax,[N]
call iprintLF
loop label
    ; переход на `label`

call quit

```

Создал и запустил исполняемый файл. Созданный цикл не принимает всех ожидаемых значений, кол-во проходов отличается от заданного в аргументе. (рис. 2.4).



```

dsdemin@fedora:~/work/arch-pc/lab08$ nano lab8-1.asm
dsdemin@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dsdemin@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dsdemin@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
5
3
1
dsdemin@fedora:~/work/arch-pc/lab08$

```

Рис. 2.4: Результат выполнения

Добавил изменение значение регистра есх в цикле.

Листинг кода:

```

;-----
; Программа вывода значений регистра 'есх'
;-----

%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:
    push ecx ; добавление значения ecx в стек
    sub ecx,1
    mov [N],ecx
    mov eax,[N]
    call iprintLF
    pop ecx ; извлечение значения ecx из стека

```

```
loop label  
; переход на `label`
```

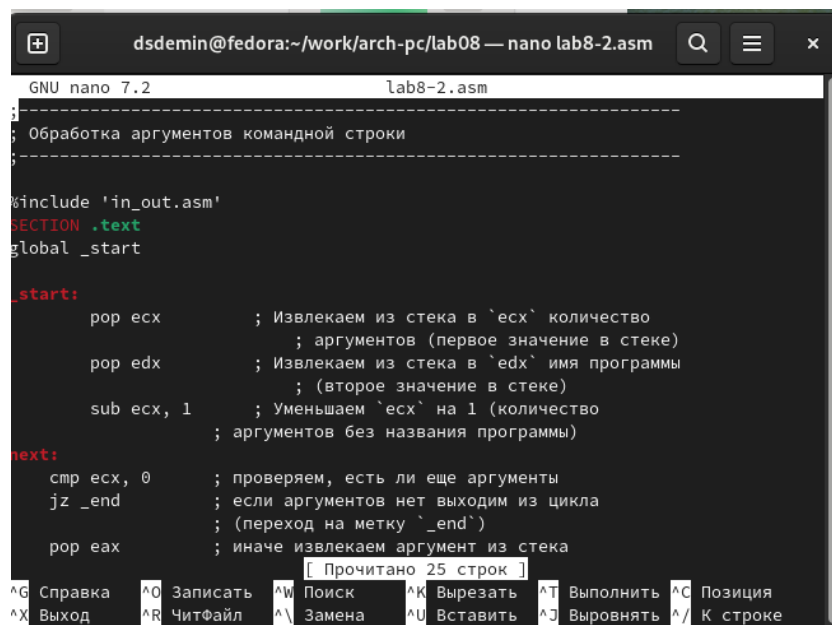
```
call quit
```

Создал и запустила исполняемый файл. Теперь регистр принимает значения с на единицу меньше значения аргумента и до 0. Число проходов цикла соответствует введенному с клавиатуры. (рис. 2.5).

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

Рис. 2.5: Результат выполнения

Создал новый файл и переписал в него код из лабораторной работы. (рис. 2.6).



```
GNU nano 7.2 lab8-2.asm  
;-----  
; 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  
  
[ Прочитано 25 строк ]  
^G Справка  ^O Записать  ^W Поиск    ^K Вырезать ^T Выполнить ^C Позиция  
^X Выход    ^R ЧитФайл  ^\ Замена   ^U Вставить ^J Выводить  ^/_ К строке
```

Рис. 2.6: Листинг кода

Листинг кода:

```

;-----
; 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

```

Создал и запустил исполняемый файл. Программой было отработано 4 аргумента (рис. 2.7).

```

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

```

Рис. 2.7: Результат выполнения

Создал новый файл и переписал в него код из лабораторной работы. рис. 2.8).

```

GNU nano 7.2 lab8-3.asm
%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

[ Прочитано 33 строки ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/_ К строке

```

Рис. 2.8: Листинг кода

Листинг кода:

```

%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

```

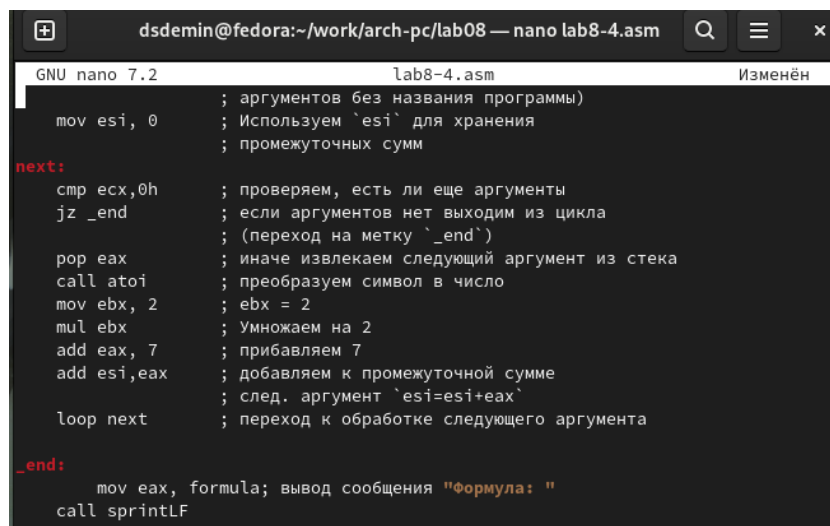
Создал исполняемый файл и запустил его. Проверил с несколькими введенными числами. (рис. 2.9).

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

Рис. 2.9: Результат работы

### 3 Выполнение самостоятельной работы

Написал программу, которая выполняет вычисления для 8 варианта задания  $f(x) = 7 + 2x$  (рис. 3.1).

A screenshot of a terminal window with the title bar 'dsdemin@fedora:~/work/arch-pc/lab08 — nano lab8-4.asm'. The window shows the GNU nano 7.2 editor with assembly code for 'lab8-4.asm'. The code includes comments in Russian explaining the logic of calculating  $f(x) = 7 + 2x$  using a loop. The code starts with 'mov esi, 0' and ends with 'call sprintf' and 'exit' instructions. The status bar at the bottom indicates 'Измeнён' (Modified).

```
GNU nano 7.2 lab8-4.asm Изменён
; 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
mov ebx, 2 ; ebx = 2
mul ebx ; Умножаем на 2
add eax, 7 ; прибавляем 7
add esi, eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента

_end:
mov eax, formula; вывод сообщения "Формула: "
call sprintf
exit
```

Рис. 3.1: Листинг кода

Запустил программу, и проверил работу с различными аргументами (рис. 3.2).

Листинг кода самостоятельной работы:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ", 0
```

```
formula db "Формула:  $f(x)=7+2x$ ",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:
```

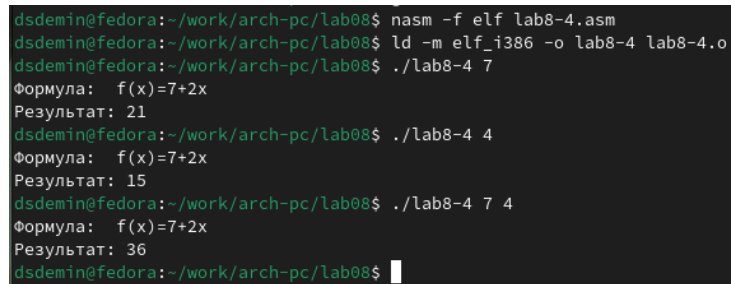
```
    cmp ecx,0h       ; проверяем, есть ли еще аргументы
    jz _end           ; если аргументов нет выходим из цикла
                     ; (переход на метку `_end`)
    pop eax           ; иначе извлекаем следующий аргумент из стека
    call atoi         ; преобразуем символ в число
    mov ebx, 2        ; ebx = 2
    mul ebx           ; Умножаем на 2
    add eax, 7        ; прибавляем 7
    add esi,eax       ; добавляем к промежуточной сумме
                     ; след. аргумент `esi=esi+eax`
    loop next         ; переход к обработке следующего аргумента
```

```
_end:
```

```
    mov eax, formula; вывод сообщения "Формула: "
```



```
call sprintf
mov eax, msg      ; вывод сообщения "Результат: "
call sprintf
mov eax, esi      ; записываем сумму в регистр `eax`
call iprintLF     ; печать результата
call quit         ; завершение программы
```



```
dsdemin@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
dsdemin@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
dsdemin@fedora:~/work/arch-pc/lab08$ ./lab8-4 7
Формула: f(x)=7+2x
Результат: 21
dsdemin@fedora:~/work/arch-pc/lab08$ ./lab8-4 4
Формула: f(x)=7+2x
Результат: 15
dsdemin@fedora:~/work/arch-pc/lab08$ ./lab8-4 7 4
Формула: f(x)=7+2x
Результат: 36
dsdemin@fedora:~/work/arch-pc/lab08$
```

Рис. 3.2: Результат работы

## **4 Выводы**

Выполнив данную лабораторную работу, я обрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.