

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Тютрюмова Анжелина

Содержание

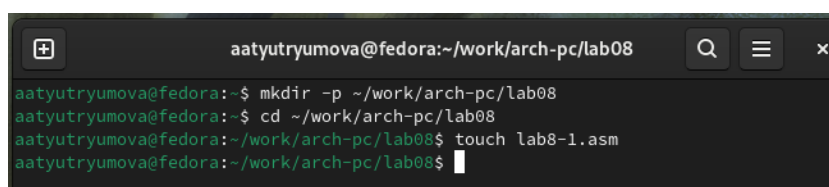
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Выполнение самостоятельной работы	15
4	Выводы	18

1 Цель работы

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

2 Выполнение лабораторной работы

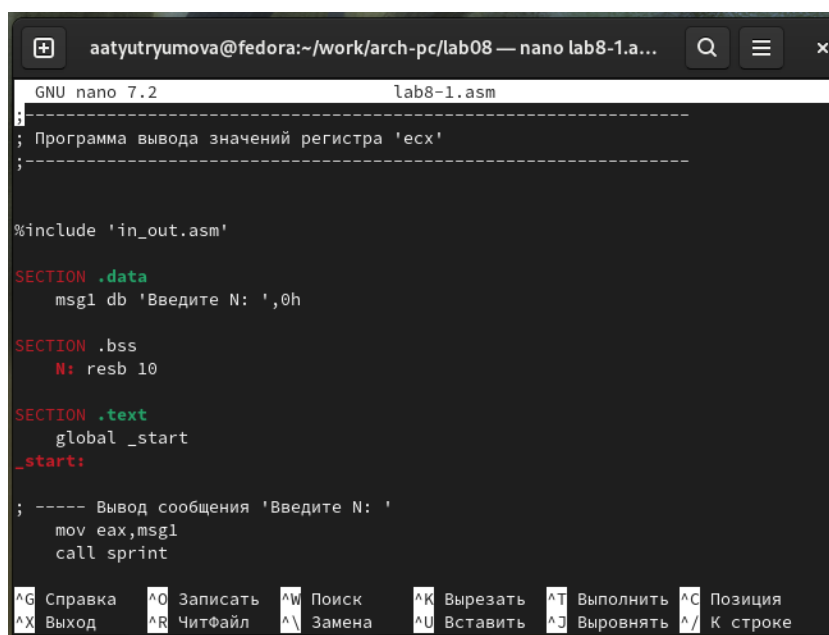
Создала и перешла в директорию для лабораторной работы создала файл lab7-8.asm (рис. 2.1).



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

Рис. 2.1: Папка для лабораторной работы

Переписала код с лабараторной работы(рис. 2.2).



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

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

Рис. 2.2: 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 исполняемый файл и запустила его. (рис. 2.3).



```

aatyutryumova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aatyutryumova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aatyutryumova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
12
11
10
9
8
7
6
5
4
3
2
1

```

Рис. 2.3: Результат выполнения

Заменяла часть кода на другой из лабараторной 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

```

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

Рис. 2.4: Результат выполнения

Добавила изменение значение регистра ecx в цикле.

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:
    push ecx ; добавление значения ecx в стек
    sub ecx,1
    mov [N],ecx

```

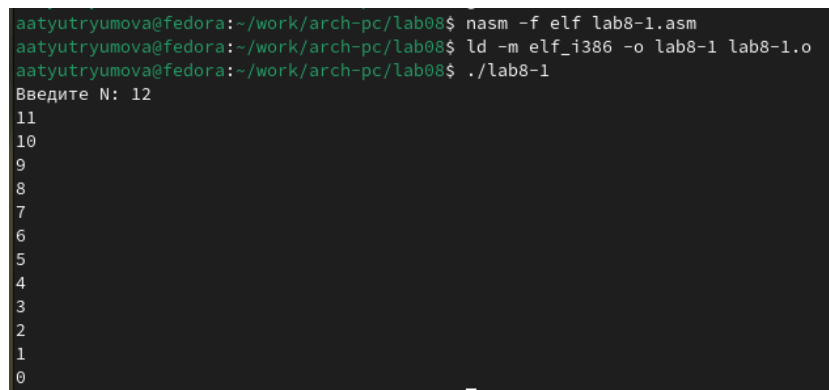
```

mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
; переход на `label`

call quit

```

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



```

aatyutryumova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aatyutryumova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aatyutryumova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
11
10
9
8
7
6
5
4
3
2
1
0

```

Рис. 2.5: Результат выполнения

Создала новый файл и переписала 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

```

Рис. 2.6: Листинг кода

Листинг кода 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:

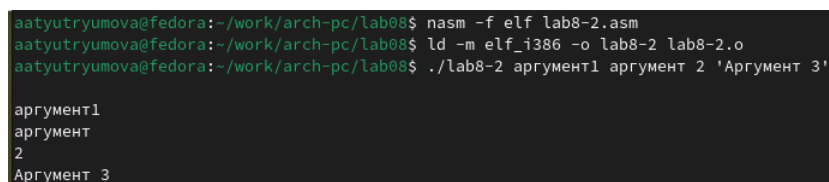
```
    cmp ecx, 0      ; проверяем, есть ли еще аргументы
    jz _end        ; если аргументов нет выходим из цикла
                    ; (переход на метку `_end`)

    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
```

Создала исполняемый файл и запустила его. Программой было отработано 4 аргумента (рис. 2.7).

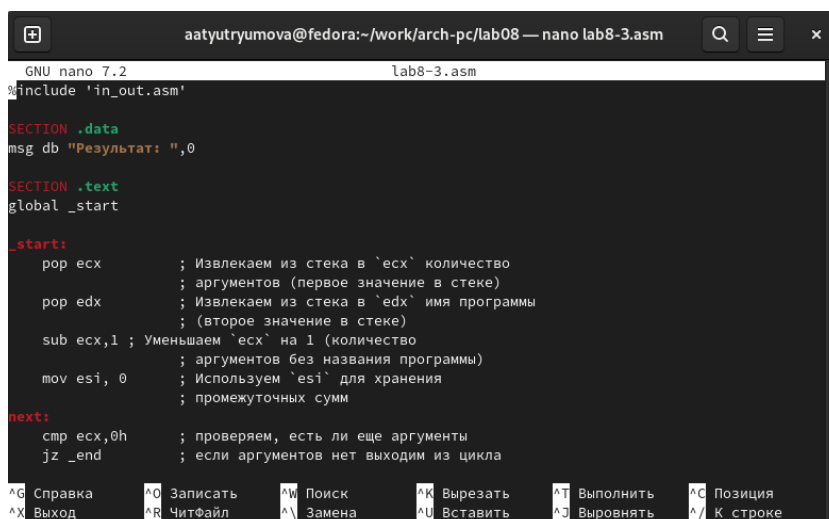


```
aatyutryumova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aatyutryumova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aatyutryumova@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'Аргумент 3'

аргумент1
аргумент
2
Аргумент 3
```

Рис. 2.7: Результат выполнения

Создала новый файл и переписала 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
```

Рис. 2.8: Листинг кода

Листинг кода 8.3:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

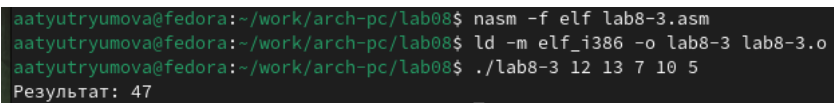
SECTION .text
global _start

_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
```

Создала исполняемый файл и запустила его. Проверила с несколькими введенными числами. (рис. 2.9).

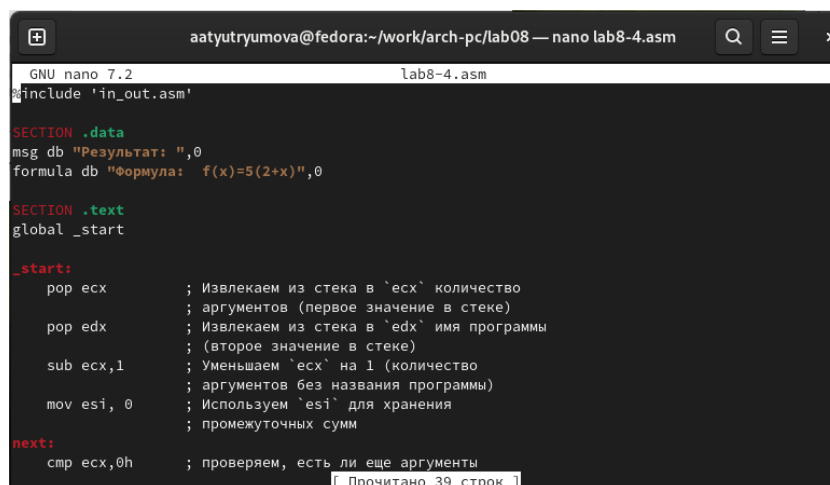


```
aatyutryumova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm  
aatyutryumova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o  
aatyutryumova@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5  
Результат: 47
```

Рис. 2.9: Результат работы

3 Выполнение самостоятельной работы

Написала программу, которая выполняет вычисления для 10 варианта задания $f(x)=5(2+x)$ (рис. 3.1).



```
GNU nano 7.2 lab8-4.asm
#include 'in_out.asm'

SECTION .data
msg db "Результат: ",0
formula db "Формула: f(x)=5(2+x)",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
```

Рис. 3.1: Листинг кода

Запустила программу, и проверила 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)=5(2+x)",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:
```

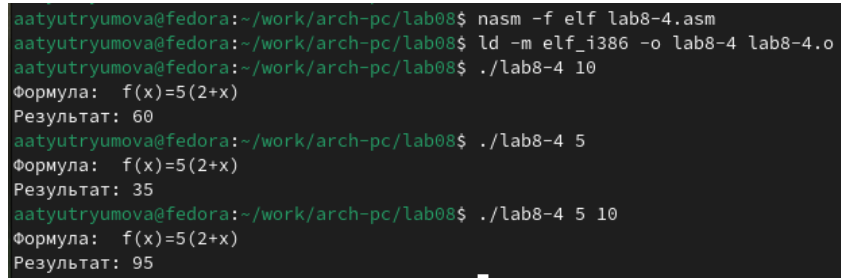
```
    cmp ecx,0h       ; проверяем, есть ли еще аргументы
    jz _end          ; если аргументов нет выходим из цикла
                     ; (переход на метку `_end`)
    pop eax          ; иначе извлекаем следующий аргумент из стека
    call atoi        ; преобразуем символ в число
    add eax, 2        ; Прибавляем 2
    mov ebx, 5        ; ebx = 5
    mul ebx          ; Умножаем на 5
    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
```



```
aatyutryumova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aatyutryumova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aatyutryumova@fedora:~/work/arch-pc/lab08$ ./lab8-4 10
Формула: f(x)=5(2+x)
Результат: 60
aatyutryumova@fedora:~/work/arch-pc/lab08$ ./lab8-4 5
Формула: f(x)=5(2+x)
Результат: 35
aatyutryumova@fedora:~/work/arch-pc/lab08$ ./lab8-4 5 10
Формула: f(x)=5(2+x)
Результат: 95
```

Рис. 3.2: Результат работы

4 Выводы

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