

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Бочаров Андрей

Содержание

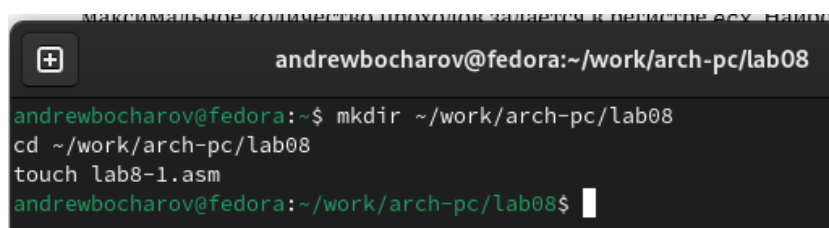
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Выполнение самостоятельной работы	13
4	Выводы	16

1 Цель работы

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

2 Выполнение лабораторной работы

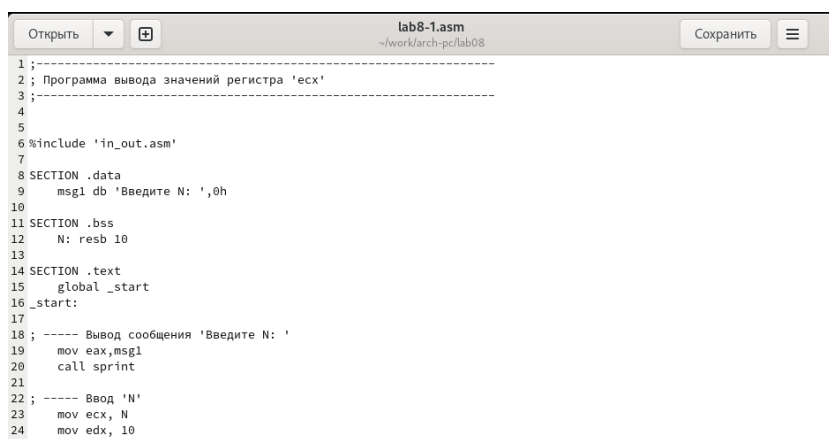
Создал и першел в каталог для 8 лабораторной работы и командой touch создал файл lab7-8.asm (рис. 2.1).



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

Рис. 2.1: Каталог lab08

Переписал код из листинга 8.1 (рис. 2.2).



```
1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4
5
6 %include 'in_out.asm'
7
8 SECTION .data
9     msg1 db 'Введите N: ',0h
10
11 SECTION .bss
12     N: resb 10
13
14 SECTION .text
15     global _start
16 _start:
17
18 ; ----- Вывод сообщения 'Введите N: '
19     mov eax,msg1
20     call sprint
21
22 ; ----- Ввод 'N'
23     mov ecx, N
24     mov edx, 10
25
```

Рис. 2.2: Листинг кода

Листинг кода 8.1:

```
;-----
; 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

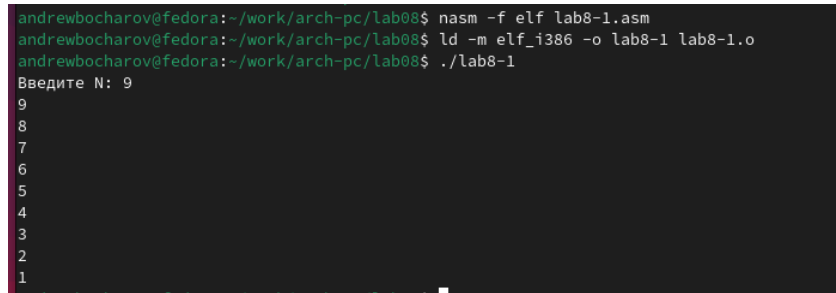
mov eax,[N]

call iprintLF ; Вывод значения `N`

loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`

call quit

Создал исполняемый файл и запустил его. (рис. 2.3).



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

Рис. 2.3: Работает

Переписал код в файле с листинга 7.2 (рис. 2.4).

```
32 ; ----- Организация цикла
33     mov ecx,[N] ; Счетчик цикла, `ecx=N`
34 label:
35     push ecx ; добавление значения ecx в стек
36     sub ecx,1
37     mov [N],ecx
38     mov eax,[N]
39     call iprintLF
40     pop ecx ; извлечение значения ecx из стека
41     loop label
42     ; переход на `label`
43
44     call quit
```

Рис. 2.4: Обновленный код

Листинг кода 8.1(Обновленный):

```
;-----  
; 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
```

```

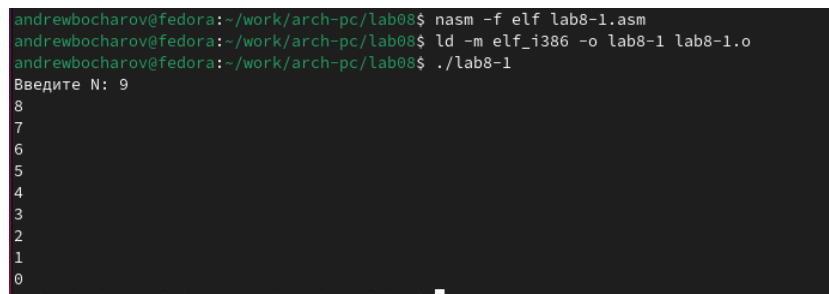
mov [N],eax

; ----- 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

```

Создал исполняемый файл и запустил его. (рис. 2.5).



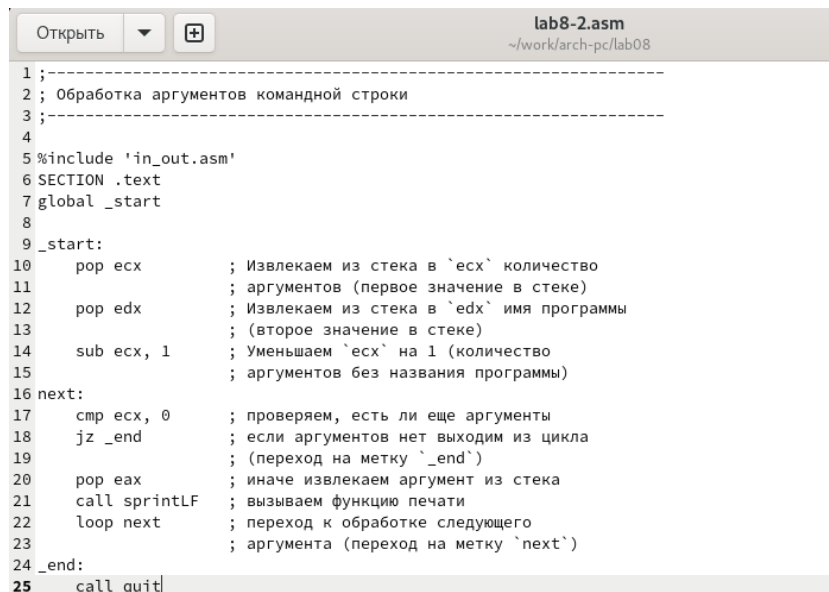
```

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

```

Рис. 2.5: Результат работы файла

Создал новый файл и переписал в него код с листинга 8.2 (рис. 2.6).



```
1 ;-----
2 ; Обработка аргументов командной строки
3 ;-----
4
5 %include 'in_out.asm'
6 SECTION .text
7 global _start
8
9 _start:
10  pop ecx          ; Извлекаем из стека в `ecx` количество
11                      ; аргументов (первое значение в стеке)
12  pop edx          ; Извлекаем из стека в `edx` имя программы
13                      ; (второе значение в стеке)
14  sub ecx, 1       ; Уменьшаем `ecx` на 1 (количество
15                      ; аргументов без названия программы)
16 next:
17  cmp ecx, 0       ; проверяем, есть ли еще аргументы
18  jz _end          ; если аргументов нет выходим из цикла
19                      ; (переход на метку `_end`)
20  pop eax          ; иначе извлекаем аргумент из стека
21  call sprintf      ; вызываем функцию печати
22  loop next        ; переход к обработке следующего
23                      ; аргумента (переход на метку `next`)
24 _end:
25  call quit
```

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
```

```

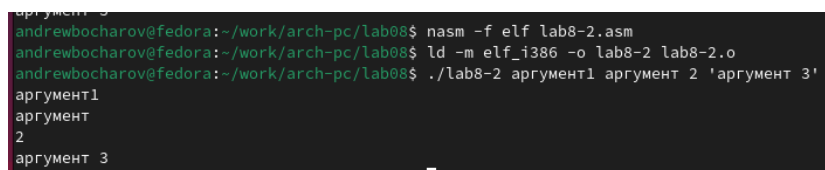
cmp ecx, 0      ; проверяем, есть ли еще аргументы
jz _end         ; если аргументов нет выходим из цикла
                ; (переход на метку `_end`)

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 исполняемый файл и запустил его. (рис. 2.7).



```

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

```

Рис. 2.7: Работает

Создал новый файл и переписал в него код с листинга 8.3 (рис. 2.8).

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5
6 SECTION .text
7 global _start
8
9 _start:
10  pop ecx          ; Извлекаем из стека в `ecx` количество
11                    ; аргументов (первое значение в стеке)
12  pop edx          ; Извлекаем из стека в `edx` имя программы
13                    ; (второе значение в стеке)
14  sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
15                    ; аргументов без названия программы)
16  mov esi, 0       ; Используем `esi` для хранения
17                    ; промежуточных сумм
18 next:
19  cmp ecx,0h       ; проверяем, есть ли еще аргументы
20  jz _end          ; если аргументов нет выходим из цикла
21                    ; (переход на метку `_end`)
22  pop eax          ; иначе извлекаем следующий аргумент из стека
23  call atoi        ; преобразуем символ в число
24  add esi,eax      ; добавляем к промежуточной сумме
25                    ; след. аргумент `esi=esi+eax`
26  loop next        ; переход к обработке следующего аргумента
27
28 _end:
29  mov eax, msg     ; вывод сообщения "Результат: "
30  call sprintf
31  mov eax, esi     ; записываем сумму в регистр `eax`
32  call iprintLF    ; печать результата
33  call quit        ; завершение программы

```

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
```

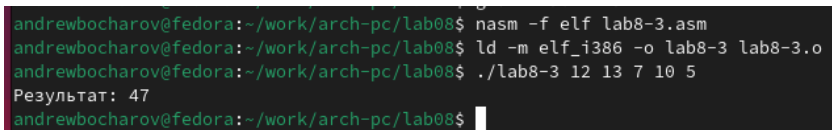
```

; (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

```

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



```

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

```

Рис. 2.9: Результат работы

3 Выполнение самостоятельной работы

Создал новый файл для самостоятельной работы, написал код, который выполняет задание соответственно варианту 13 (рис. 3.1).

```
19 next:
20     cmp ecx,0h      ; проверяем, есть ли еще аргументы
21     jz _end         ; если аргументов нет выходим из цикла
22                     ; (переход на метку `_end`)
23     pop eax         ; иначе извлекаем следующий аргумент из стека
24     call atoi        ; преобразуем символ в число
25     mov ebx, 12      ; ebx = 12
26     mul ebx          ; Умножаем на 12
27     sub eax, 7       ; вычитаем 7
28     add esi,eax      ; добавляем к промежуточной сумме
29                     ; след. аргумент `esi=esi+eax`
30     loop next        ; переход к обработке следующего аргумента
31
32 _end:
33     mov eax, formula; вывод сообщения "Формула: "
34     call sprintf
35     mov eax, msg      ; вывод сообщения "Результат: "
36     call sprintf
37     mov eax, esi       ; записываем сумму в регистр `eax`
38     call iprintLF      ; печать результата
39     call quit          ; завершение программы
```

Рис. 3.1: Листинг кода

Запустил код, и проверил с различными аргументами командной строки (рис. 3.2).

Листинг кода самостоятельной работы:

```
%include 'in_out.asm'
```

SECTION .data

msg db "Результат: ",0

formula db "Формула: $f(x)=12x-7$ ",0

SECTION .text

global _start

_start:

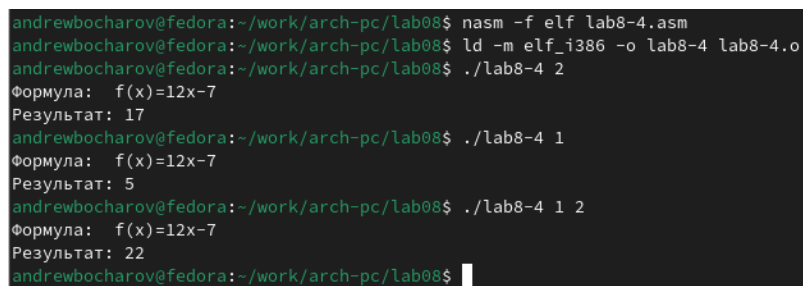
```
    pop ecx          ; Извлекаем из стека в `ecx` количество
                     ; аргументов (первое значение в стеке)
    pop edx          ; Извлекаем из стека в `edx` имя программы
                     ; (второе значение в стеке)
    sub ecx,1        ; Уменьшаем `ecx` на 1 (количество
                     ; 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, 12       ; ebx = 12
    mul ebx          ; Умножаем на 12
    sub eax, 7        ; вычитаем 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 sprintfLF  
mov eax, msg      ; вывод сообщения "Результат: "  
call sprintf  
mov eax, esi      ; записываем сумму в регистр `eax`  
call iprintLF     ; печать результата  
call quit         ; завершение программы
```



```
andrewbocharov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm  
andrewbocharov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o  
andrewbocharov@fedora:~/work/arch-pc/lab08$ ./lab8-4 2  
Формула: f(x)=12x-7  
Результат: 17  
andrewbocharov@fedora:~/work/arch-pc/lab08$ ./lab8-4 1  
Формула: f(x)=12x-7  
Результат: 5  
andrewbocharov@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2  
Формула: f(x)=12x-7  
Результат: 22  
andrewbocharov@fedora:~/work/arch-pc/lab08$
```

Рис. 3.2: Листинг кода

4 Выводы

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