

# **Лабораторная работа №9. Программирование цикла. Обработка**

**Дисциплина: Архитектура ЭВМ**

Плескачева Елизавета Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация циклов . . . . .	6
2.2	Обработка аргументов командной строки . . . . .	9
2.2.1	Выведение суммы аргументов . . . . .	10
2.2.2	Выведение произведения аргументов . . . . .	10
<b>3</b>	<b>Задания для самостоятельной работы</b>	<b>12</b>
<b>4</b>	<b>Выводы</b>	<b>15</b>

# Список иллюстраций

2.1	Создание папки и файла . . . . .	6
2.2	Листинг 9.1 в lab9-1.asm . . . . .	7
2.3	Компиляция и запуск lab9-1 . . . . .	7
2.4	Добавление строки . . . . .	8
2.5	Измененный вывод программы . . . . .	8
2.6	Изменение кода lab9-1.asm . . . . .	8
2.7	Вывод измененной програмы . . . . .	9
2.8	Текст в lab9-2.asm . . . . .	9
2.9	Запуск lab9-2 с аргументами . . . . .	10
2.10	Запуск кода lab9-3 . . . . .	10
2.11	Измененная часть программы . . . . .	10
2.12	Запуск кода lab9-3-mult . . . . .	11
3.1	Программа выводящая сумму результатов функций . . . . .	13
3.2	Вывод программы lab9-4 . . . . .	14

## Список таблиц

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработка аргументов командной строки.

## 2 Выполнение лабораторной работы

### 2.1 Реализация циклов

Создадим каталог для выполнения лабораторной, перейдем в него

```
[eapleskacheva@localhost ~]$ cd  
[eapleskacheva@localhost ~]$ mkdir ~/work/arch-pc/lab09  
[eapleskacheva@localhost ~]$ cd ~/work/arch-pc/lab09  
[eapleskacheva@localhost lab09]$ touch lab9-1.asm  
[eapleskacheva@localhost lab09]$
```

Рис. 2.1: Создание папки и файла

введем листинг 9.1 в файл

```

1 %include 'in_out.asm'
2
3 SECTION .data
4
5 msg1 db 'Введите N: ',0h
6
7 SECTION .bss
8
9 N: resb 10
10
11 SECTION .text
12
13 global _start
14
15 _start:
16 ; ----- Вывод сообщения 'Введите N: '
17 mov eax,msg1
18 call sprint
19 ; ----- Ввод 'N'
20 mov ecx, N
21 mov edx, 10
22 call sread
23 ; ----- Преобразование 'N' из символа в число
24 mov eax,N
25 call atoi
26 mov [N],eax
27 ; ----- Организация цикла
28 mov ecx,[N] ; Счетчик цикла, `ecx=N`
29 label:
30
31 mov [N],ecx
32 mov eax,[N]
33 call iprintLF ; Вывод значения `N`
34
35 loop label ; `ecx=ecx-1` и если `ecx` не `0`
36 ; переход на `label`
37 call quit
38

```

Рис. 2.2: Листинг 9.1 в lab9-1.asm

Скомпилируем и запустим код. Проверим его.

```

[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-1.asm
ld -m elf_i386 -o ./lab9-1 ./lab9-1.o
./lab9-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
[eapleskacheva@localhost lab09]$ 

```

Рис. 2.3: Компиляция и запуск lab9-1

Программа выводит числа от 10 до 1

Добавим строчку `sub ecx, 1` в программу

```

30
31     sub ecx, 1
32     mov [N],ecx
33     mov eax,[N]

```

Рис. 2.4: Добавление строки

Скомпилируем и запустим.

```

[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-1.asm
ld -m elf_i386 -o ./lab9-1 ./lab9-1.o
./lab9-1
Введите N: 10
9
7
5
3
1
[eapleskacheva@localhost lab09]$

```

Рис. 2.5: Измененный вывод программы

Теперь программа выводит числа 9 7 5 3 1

Что бы программа работала как раньше, но начинала с 9, обернем код в push  
pop

```

29 label:
30
31     push ecx
32     sub ecx, 1
33     mov [N],ecx
34     mov eax,[N]
35     call iprintLF ; Вывод значения `N`
36     pop ecx

```

Рис. 2.6: Изменение кода lab9-1.asm

Скомпилируем и запустим



```

[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-1.asm
ld -m elf_i386 -o ./lab9-1 ./lab9-1.o
./lab9-1

Введите N: 10
9
8
7
6
5
4
3
2
1
0
[eapleskacheva@localhost lab09]$ 

```

Рис. 2.7: Вывод измененной программы

Теперь программа выводит числа от 9 до 0

## 2.2 Обработка аргументов командной строки

Введем в lab9-2.asm листинг 9.2

```

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

```

Рис. 2.8: Текст в lab9-2.asm

Скомпилируем и запустим программу указав аргументы

```
[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-2.asm
ld -m elf_i386 -o ./lab9-2 ./lab9-2.o
./lab9-2
[eapleskacheva@localhost lab09]$ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[eapleskacheva@localhost lab09]$
```

Рис. 2.9: Запуск lab9-2 с аргументами

Программа Выводит аргументы последовательно на экран. Программа выводит 4 аргумента

### 2.2.1 Выведение суммы аргументов

Введем код из листинга 9.3 в lab9-3, скомпилируем и запустим его

Введем аргументы 12 13 7 10 5

```
[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-3.asm
ld -m elf_i386 -o ./lab9-3 ./lab9-3.o
./lab9-3 12 13 7 10 5
Результат: 47
[eapleskacheva@localhost lab09]$
```

Рис. 2.10: Запуск кода lab9-3

Программа вывела сумму аргументов - 47

### 2.2.2 Выведение произведения аргументов

Изменим программу lab9-3.asm так, что бы она умножала аргументы. Сохраним измененную прогармму в lab9-3-mult.asm

```
[eapleskacheva@localhost lab9]$ ./lab9-3-mult 1 2 3 4
Результат: 24
[eapleskacheva@localhost lab9]$ ./lab9-3-mult 1 2 3 4 5
Результат: 120
[eapleskacheva@localhost lab9]$
```

Рис. 2.11: Измененная часть программы

Скомпилируем и запустим программу, проверим ее на несокльких аргументах

```

21
22 push edx
23 push eax
24 mov edx, eax
25 mov eax, esi
26
27 mul edx ; добавляем к промежуточной сумме
28 mov esi, eax
29
30 pop eax
31 pop edx
32 ; след. аргумент `esi=esi+eax`
33 loop next ; переход к обработке следующего аргумента
34

```

Рис. 2.12: Запуск кода lab9-3-mult

Программа выводит правильный результат

### 3 Задания для самостоятельной работы

Мой вариант - 2, поэтому пишем программу, которая выводит сумму результатов функции  $f(x) = 3x - 1$

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 func db "f(x) = 3x - 1: ",0
5 msg db "Результат: ",0
6 SECTION .text
7
8 global _start
9
10 _start:
11     pop ecx ; Извлекаем из стека в `ecx` количество
12             ; аргументов (первое значение в стеке)
13     pop edx ; Извлекаем из стека в `edx` имя программы
14             ; (второе значение в стеке)
15     sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
16             ; аргументов без названия программы)
17     mov esi, 0 ; Используем `esi` для хранения
18             ; промежуточных сумм
19     mov eax, func
20
21     call sprintf
22
23
24 next:
25     cmp ecx,0h ; проверяем, есть ли еще аргументы
26     jz _end ; если аргументов нет выходим из цикла
27             ; (переход на метку `_end`)
28     pop eax ; иначе извлекаем следующий аргумент из стека
29     call atoi ; преобразуем символ в число
30
31
32
33
34     push ecx
35
36     mov ecx,3
37     mul ecx
38     dec eax
39
40     pop ecx
41
42
43
44
45
46
47     add esi,eax ; добавляем к промежуточной сумме
48
49     loop next ; переход к обработке следующего аргумента
50
51 _end:
52
53     mov eax, msg ; вывод сообщения "Результат: "
54     call sprint
55     mov eax, esi ; записываем сумму в регистр `eax`
56     call iprintLF ; печать результата
57
58     call quit ; завершение программы
59

```

Рис. 3.1: Программа выводящая сумму результатов функций

Запустим программу и проверим ее на аргументах 1 2 3 4

```
[eapleskacheva@localhost lab09]$ ./lab9-4 1 2 3 4  
f(x) = 3x - 1:  
Результат: 26  
[eapleskacheva@localhost lab09]$
```

Рис. 3.2: Вывод программы lab9-4

Результат верный

## 4 Выводы

Мы приобрели навыки написания программ с использованием циклов и научились обрабатывать аргументы командной строки.