

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Турсунбоев Сардорбек

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	22

Список иллюстраций

3.1	Код программы lab8-1.asm	8
3.2	Тестирование программы lab8-1.asm	9
3.3	Код программы lab8-1.asm	10
3.4	Тестирование программы lab8-1.asm	11
3.5	Код программы lab8-1.asm	12
3.6	Тестирование программы lab8-1.asm	13
3.7	Код программы lab8-2.asm	14
3.8	Тестирование программы lab8-2.asm	15
3.9	Код программы lab8-3.asm	16
3.10	Тестирование программы lab8-3.asm	17
3.11	Код программы lab8-3.asm	18
3.12	Тестирование программы lab8-3.asm	19
3.13	Код программы lab8-4.asm	20
3.14	Тестирование программы lab8-4.asm	21

Список таблиц

1 Цель работы

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

2 Задание

1. Изучить операторы цикла
2. Изучить организацию стека
3. Изучить передачу аргументов
4. Рассмотреть примеры программ
5. Выполнить самостоятельное задание

3 Выполнение лабораторной работы

Я создал каталог для программ лабораторной работы No8 и перешел в него. Далее создал файл lab8-1.asm.

Я написал в файле lab8-1.asm текст программы из листинга 8.1. После этого создал исполняемый файл и проверил его работу.

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

Рис. 3.1: Код программы lab8-1.asm


```
[sktursunboev@fedora lab08]$ touch lab8-1.asm
[sktursunboev@fedora lab08]$ nasm -f elf lab8-1.asm
[sktursunboev@fedora lab08]$ ld -m elf_i386 lab8-1.o -o lab8-1
[sktursunboev@fedora lab08]$ ./lab8-1
Введите N: 3
3
2
1
[sktursunboev@fedora lab08]$ ./lab8-1
Введите N: 4
4
3
2
1
[sktursunboev@fedora lab08]$
```

Рис. 3.2: Тестирование программы lab8-1.asm

Этот пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы.

Я внес изменения в текст программы, добавив изменение значения регистра `ecx` в цикле. Затем создал исполняемый файл и проверил его работу. Программа запускала бесконечный цикл при нечетном значении `N` и выводила только нечетные числа при четном `N`.

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

Рис. 3.3: Код программы lab8-1.asm

```
4294922284
4294922282
4294922280
4294922278
4294922276
4294922274
4294922272
4294922270
429^C
[sktursunboev@fedora lab08]$ ./lab8-1
Введите N: 4
3
1
[sktursunboev@fedora lab08]$
```

Рис. 3.4: Тестирование программы lab8-1.asm

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Я внес изменения в текст программы, добавив команды `push` и `pop` для сохранения значения счетчика цикла `loop`. После этого создал исполняемый файл и проверил его работу. Теперь программа выводила числа от $N-1$ до 0, и число проходов цикла соответствовало значению N .

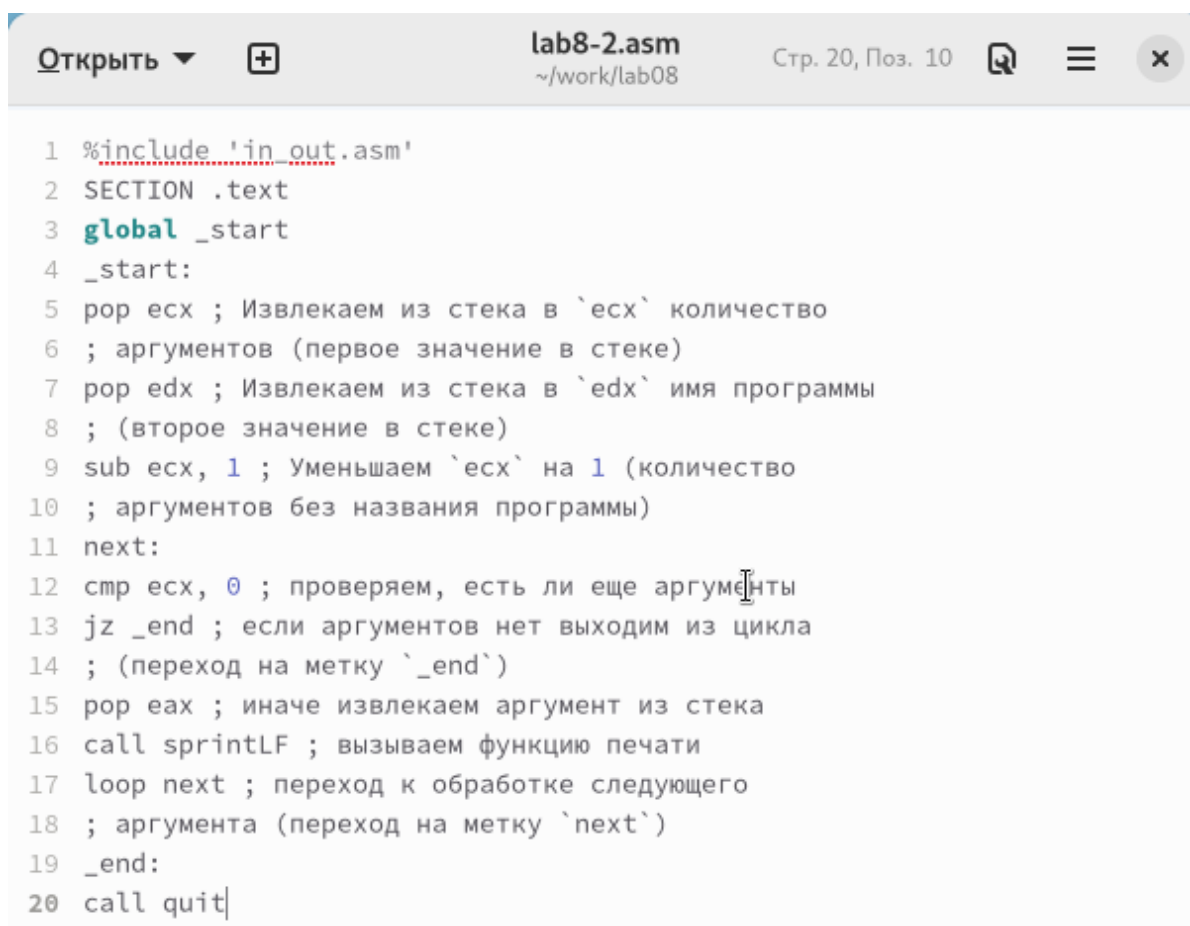
```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование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

Рис. 3.5: Код программы lab8-1.asm

```
[sktursunboev@fedora lab08]$ nasm -f elf lab8-1.asm
[sktursunboev@fedora lab08]$ ld -m elf_i386 lab8-1.o -o lab8-1
[sktursunboev@fedora lab08]$ ./lab8-1
Введите N: 4
3
2
1
0
[sktursunboev@fedora lab08]$ ./lab8-1
Введите N: 3
2
1
0
[sktursunboev@fedora lab08]$
```

Рис. 3.6: Тестирование программы lab8-1.asm

Я создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2. Затем создал исполняемый файл, запустил его и указал 5 аргументов. Программа успешно обработала эти 5 аргументов.



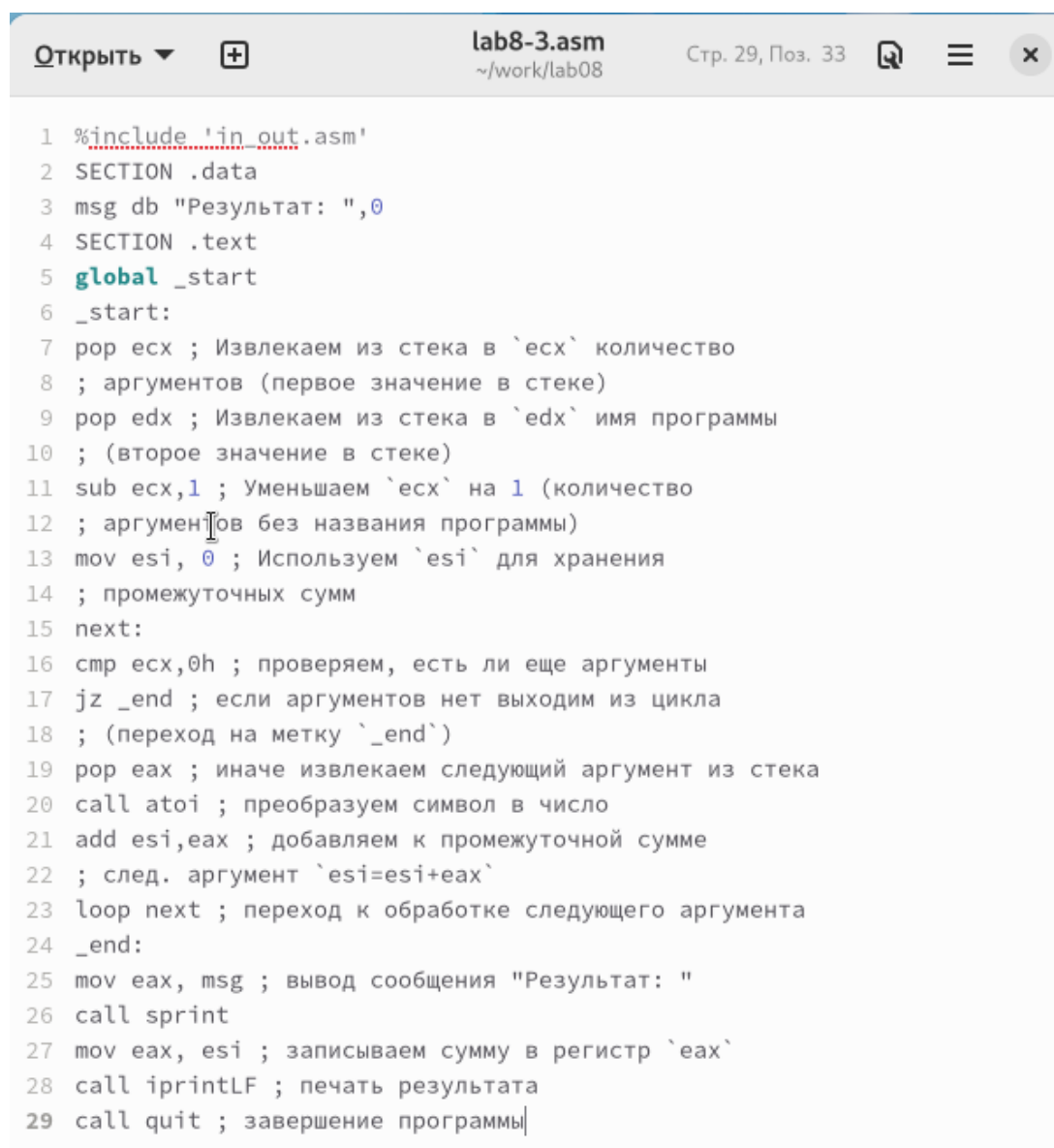
```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов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

Рис. 3.7: Код программы lab8-2.asm

```
[sktursunboev@fedora lab08]$  
[sktursunboev@fedora lab08]$ touch lab8-2.asm  
[sktursunboev@fedora lab08]$  
[sktursunboev@fedora lab08]$ nasm -f elf lab8-2.asm  
[sktursunboev@fedora lab08]$ ld -m elf_i386 lab8-2.o -o lab8-2  
[sktursunboev@fedora lab08]$ ./lab8-2 1 2 3 4  
1  
2  
3  
4  
[sktursunboev@fedora lab08]$ ./lab8-2 argument 1 argument 2 'argument 3'  
argument  
1  
argument  
2  
argument 3  
[sktursunboev@fedora lab08]$
```

Рис. 3.8: Тестирование программы lab8-2.asm

Рассмотрим ещё один пример программы, которая выводит сумму чисел, переданных в неё в качестве аргументов.



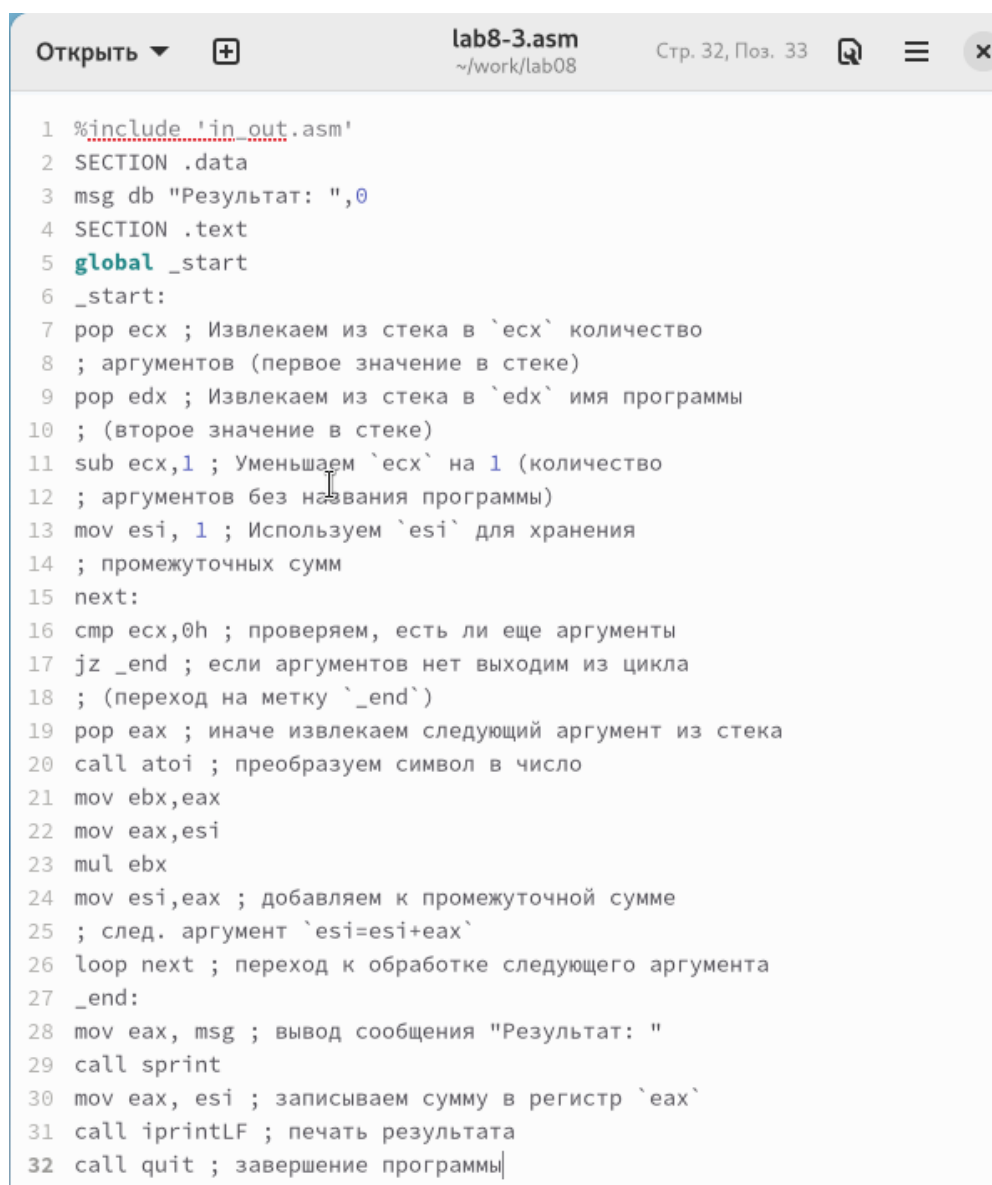
```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ;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

Рис. 3.9: Код программы lab8-3.asm


```
[sktursunboev@fedora lab08]$  
[sktursunboev@fedora lab08]$ touch lab8-3.asm  
[sktursunboev@fedora lab08]$ nasm -f elf lab8-3.asm  
[sktursunboev@fedora lab08]$ ld -m elf_i386 lab8-3.o -o lab8-3  
[sktursunboev@fedora lab08]$ ./lab8-3 1 2 3 4 5 6  
Результат: 21  
[sktursunboev@fedora lab08]$ ./lab8-3 1 2 3  
Результат: 6  
[sktursunboev@fedora lab08]$
```

Рис. 3.10: Тестирование программы lab8-3.asm

Я внёс изменения в текст программы из листинга 8.3, чтобы она вычисляла произведение аргументов командной строки.



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

Рис. 3.11: Код программы lab8-3.asm

```

[sktursunboev@fedora lab08]$
[sktursunboev@fedora lab08]$ touch lab8-3.asm
[sktursunboev@fedora lab08]$ nasm -f elf lab8-3.asm
[sktursunboev@fedora lab08]$ ld -m elf_i386 lab8-3.o -o lab8-3
[sktursunboev@fedora lab08]$ ./lab8-3 1 2 3 4 5 6
Результат: 21
[sktursunboev@fedora lab08]$ ./lab8-3 1 2 3
Результат: 6
[sktursunboev@fedora lab08]$ nasm -f elf lab8-3.asm
[sktursunboev@fedora lab08]$ ld -m elf_i386 lab8-3.o -o lab8-3
[sktursunboev@fedora lab08]$ ./lab8-3 1 2 3 4 5 6
Результат: 720
[sktursunboev@fedora lab08]$ ./lab8-3 1 2 3
Результат: 6
[sktursunboev@fedora lab08]$

```

Рис. 3.12: Тестирование программы lab8-3.asm

Написал программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создал исполняемый файл и проверил его работу на нескольких наборах x .

для варианта 4 $f(x) = 2(x - 10)$



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 2(x - 10)',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 sub eax,10
22 mov ebx,2
23 mul ebx
24 add esi,eax
25
26 loop next
27
28 _end:
29 mov eax, msg
30 call sprint
31 mov eax, esi
32 call iprintLF
33 call quit
```

Рис. 3.13: Код программы lab8-4.asm

```
[sktursunboev@fedora lab08]$ touch lab8-4.asm
[sktursunboev@fedora lab08]$ nasm -f elf lab8-4.asm
[sktursunboev@fedora lab08]$ ld -m elf_i386 lab8-4.o -o lab8-4
[sktursunboev@fedora lab08]$ ./lab8-4 10
f(x)= 2(x - 10)
Результат: 0
[sktursunboev@fedora lab08]$ ./lab8-4 11
f(x)= 2(x - 10)
Результат: 2
[sktursunboev@fedora lab08]$ ./lab8-4 11 12 13 14 15 16
f(x)= 2(x - 10)
Результат: 42
[sktursunboev@fedora lab08]$
```

Рис. 3.14: Тестирование программы lab8-4.asm

4 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере `naasm`.