

# **Отчёт по лабораторной работе 8**

**Архитектура компьютера**

Албаша Али Омран

# **Содержание**

|   |           |
|---|-----------|
| <b>1 Цель работы</b>                    | <b>5</b>  |
| <b>2 Выполнение лабораторной работы</b> | <b>6</b>  |
| 2.1 Реализация циклов в NASM . . . . .  | 6         |
| 2.2 Самостоятельное задание . . . . .   | 16        |
| <b>3 Выводы</b>                         | <b>19</b> |

# **Список иллюстраций**

|  |    |
|--|----|
| 2.1 Создан каталог . . . . .                   | 6  |
| 2.2 Программа lab8-1.asm . . . . .             | 7  |
| 2.3 Запуск программы lab8-1.asm . . . . .      | 8  |
| 2.4 Программа lab8-1.asm . . . . .             | 9  |
| 2.5 Запуск программы lab8-1.asm . . . . .      | 10 |
| 2.6 Программа lab8-1.asm . . . . .             | 11 |
| 2.7 Запуск программы lab8-1.asm . . . . .      | 12 |
| 2.8 Программа lab8-2.asm . . . . .             | 13 |
| 2.9 Запуск программы lab8-2.asm . . . . .      | 13 |
| 2.10 Программа lab8-3.asm . . . . .            | 14 |
| 2.11 Запуск программы lab8-3.asm . . . . .     | 14 |
| 2.12 Программа lab8-3.asm . . . . .            | 15 |
| 2.13 Запуск программы lab8-3.asm . . . . .     | 16 |
| 2.14 Программа lab8-task1.asm . . . . .        | 17 |
| 2.15 Запуск программы lab8-task1.asm . . . . . | 18 |

# **Список таблиц**

# **1 Цель работы**

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

## 2 Выполнение лабораторной работы

### 2.1 Реализация циклов в NASM

Создал каталог для программ лабораторной работы № 8 и файл lab8-1.asm (рис. 2.1).

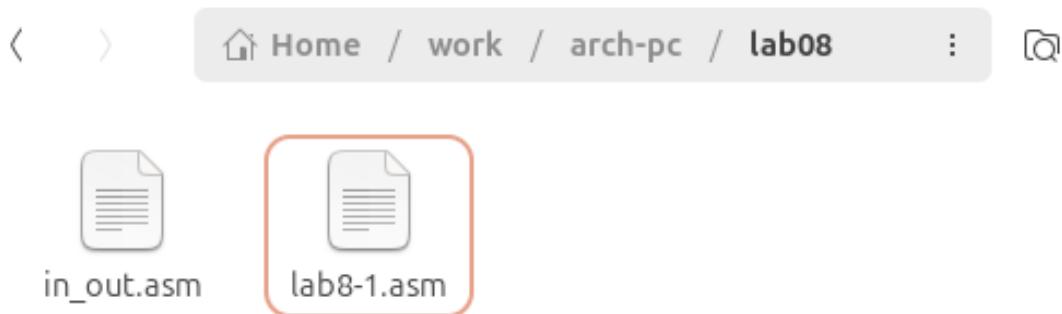


Рисунок 2.1: Создан каталог

При реализации циклов в NASM с использованием инструкции `loop` необходимо помнить о том, что эта инструкция использует регистр `esi` в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра `esi`.

Написал в файл `lab8-1.asm` текст программы из листинга 8.1 (рис. 2.2). Создал исполняемый файл и проверил его работу (рис. 2.3).

```
lab8-1.asm +1
home > aoalbasha > work > arch-pc > lab08 > lab8-1.asm
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`           I
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
```

Рисунок 2.2: Программа lab8-1.asm

```
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 3
3
2
1
aoalbasha@VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 2.3: Запуск программы lab8-1.asm

Данный пример показывает, что использование регистра `есх` в теле цикла `loop` может привести к некорректной работе программы. Изменил текст программы, добавив изменение значения регистра `есх` в цикле (рис. 2.4). Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N` (рис. 2.5).

The screenshot shows a code editor window with the file 'lab8-1.asm' open. The file path is 'home > aoalbasha > work > arch-pc > lab08 > lab8-1.asm'. The code is written in assembly language and includes comments in Russian. The code reads a value from the user, converts it to an integer, and then prints it out N times.

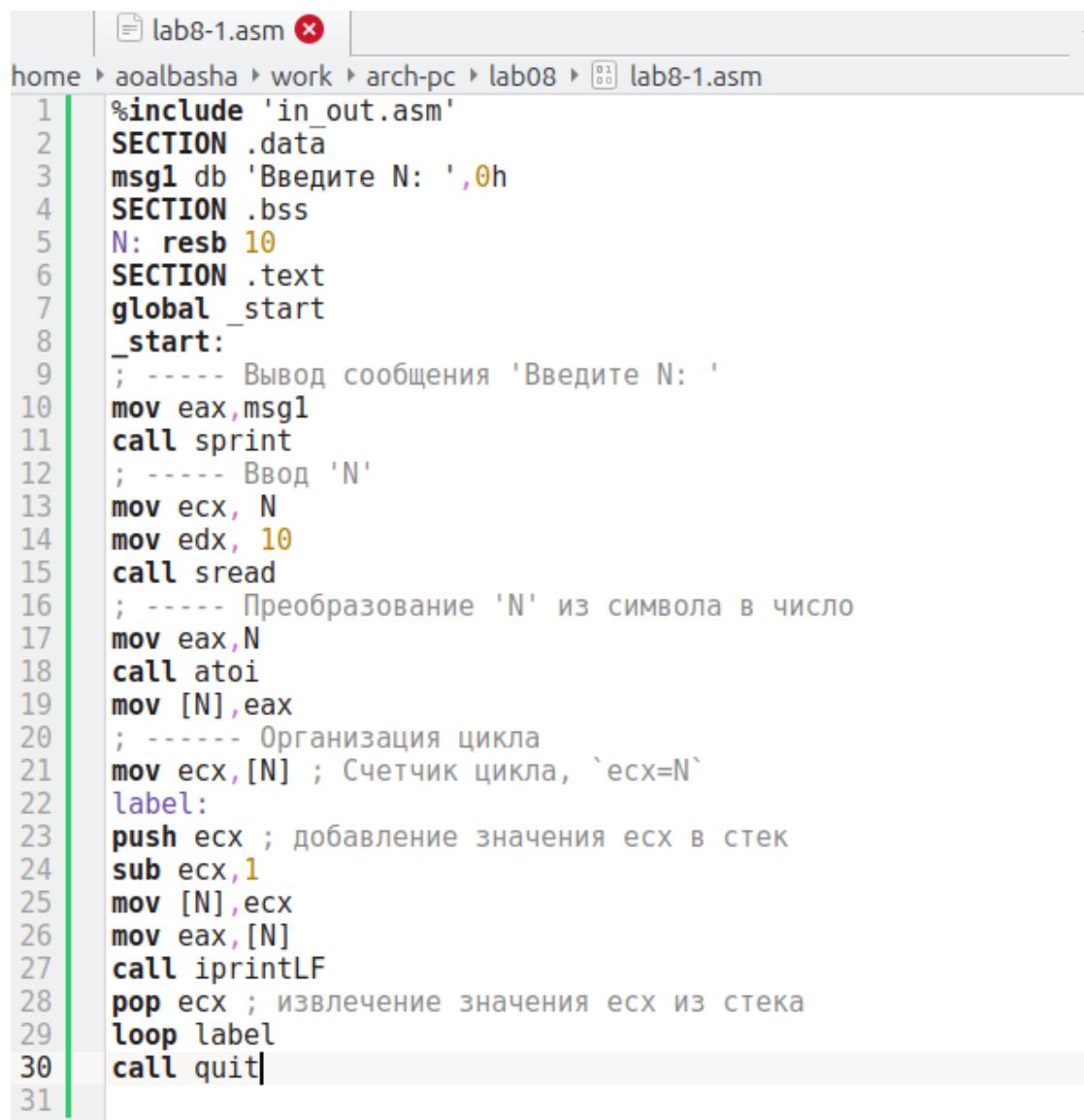
```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование '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
30
```

Рисунок 2.4: Программа lab8-1.asm

```
4294751558
4294751556
4294751554
4294751552
4294751550
4294751548
4294751546
42947^C
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
aoalbasha@VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 2.5: Запуск программы lab8-1.asm

Для корректной работы программы с использованием регистра `esx` можно применить стек. Внес изменения в программу, добавив команды `push` и `pop` для сохранения значения счетчика цикла `loop` (рис. 2.6). Создал исполняемый файл и проверил его работу (рис. 2.7). Программа выводит числа от  $N-1$  до 0, количество проходов цикла соответствует  $N$ .



The screenshot shows a Windows Notepad window with the file name "lab8-1.asm" in the title bar. The window displays assembly language code. The code includes directives like %include, SECTION, and global \_start. It also contains instructions for outputting a message, reading input, converting input to a number, and performing a loop operation. The assembly code is color-coded, with labels in blue, registers in green, and memory addresses in red.

```
%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
call quit
```

Рисунок 2.6: Программа lab8-1.asm

```
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 3
2
1
0
aoalbasha@VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 2.7: Запуск программы lab8-1.asm

Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и написал в него текст программы из листинга 8.2 (рис. 2.8). Скомпилировал исполняемый файл и запустил его, указав аргументы. Программа обработала 4 аргумента. Аргументами считаются слова или числа, разделенные пробелом (рис. 2.9).

```
lab8-2.asm
```

```
home > aoalbasha > work > arch-pc > lab08 > lab8-2.asm
```

```
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 sprintLF ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit|
```

Рисунок 2.8: Программа lab8-2.asm

```
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-2
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 argument1 argument 2 'argume
nt 3'
argument1
argument
2
argument 3
aoalbasha@VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 2.9: Запуск программы lab8-2.asm

Рассмотрим еще один пример программы, которая выводит сумму чисел, передаваемых в программу как аргументы (рис. 2.10, рис. 2.11).

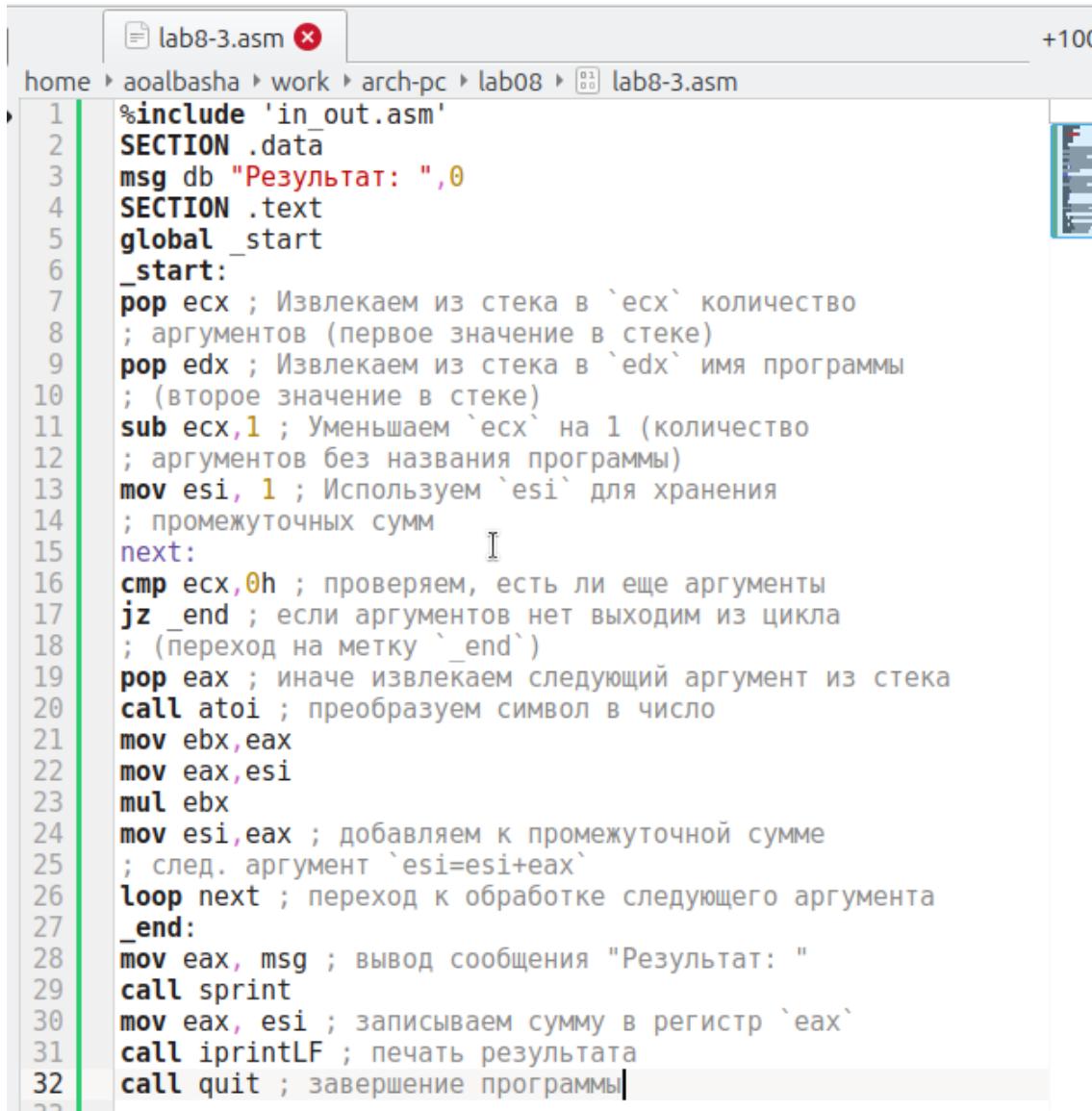
```
lab8-3.asm
home > aoalbasha > work > arch-pc > lab08 > lab8-3.asm
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `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
```

Рисунок 2.10: Программа lab8-3.asm

```
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 4 5
Результат: 9
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 4 5 6 7 8 9
Результат: 39
aoalbasha@VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 2.11: Запуск программы lab8-3.asm

Изменил текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 2.12, рис. 2.13).



The screenshot shows a Windows Notepad window with the file name "lab8-3.asm" in the title bar. The file path is displayed above the title bar: "home > aoalbasha > work > arch-pc > lab08 > lab8-3.asm". The code itself is written in assembly language:

```
%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, 1 ; 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,eax
    mov eax,esi
    mul ebx
    mov 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|
```

Рисунок 2.12: Программа lab8-3.asm

```
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 4 5
Результат: 20
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 4 5 6 7 8 9
Результат: 60480
aoalbasha@VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 2.13: Запуск программы lab8-3.asm

## 2.2 Самостоятельное задание

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  $f(x)$  выбирается в соответствии с вариантом, полученным при выполнении лабораторной работы № 7.

Создал исполняемый файл и проверил его работу на нескольких наборах  $x$  (рис. 2.14, рис. 2.15).

Для варианта 11:

$$f(x) = 15x + 2$$

The screenshot shows a code editor window with the file `lab8-task1.asm` open. The file path is `home > aoalbasha > work > arch-pc > lab08 > lab8-task1.asm`. The assembly code is as follows:

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 15x + 2',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 mov ebx,15
22 mul ebx
23 add eax,2|
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
34
```

Рисунок 2.14: Программа lab8-task1.asm

```
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-task1.asm
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-task1.o -o lab8-task1
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-task1
f(x)= 15x + 2
Результат: 0
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-task1 0
f(x)= 15x + 2
Результат: 2
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-task1 1
f(x)= 15x + 2
Результат: 17
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ ./lab8-task1 1 3 4 6 5
f(x)= 15x + 2
Результат: 295
aoalbasha@VirtualBox:~/work/arch-pc/lab08$ █
```

Рисунок 2.15: Запуск программы lab8-task1.asm

Убедился, что программа считает правильно  $f(0) = 2$ ,  $f(1) = 17$ .

## **3 Выводы**

Освоили работы со стеком, циклом и аргументами на ассемблере nasm.