Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютера

Адмиральская Александра Андреевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1.Реализация циклов в NASM 2.Обработка аргументов командной строки 3.Вычисление суммы аргументов командной строки 4.Выполнение задания для самостоятельной работы

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

4 Выполнение лабораторной работы

Для начала создаем каталог для программам лабораторной работы № 8, переходим в него и создаем файл lab8-1.asm (рис. 4.1).

```
aaadmiraljskaya@dk3n55 ~ $ mkdir ~/work/arch-pc/lab08
aaadmiraljskaya@dk3n55 ~ $ cd ~/work/arch-pc/lab08
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 4.1: Создание каталога и файла

Открываем созданной файл командой mc и вводим в него текст программы из листинга 8.1 (рис. 4.2).

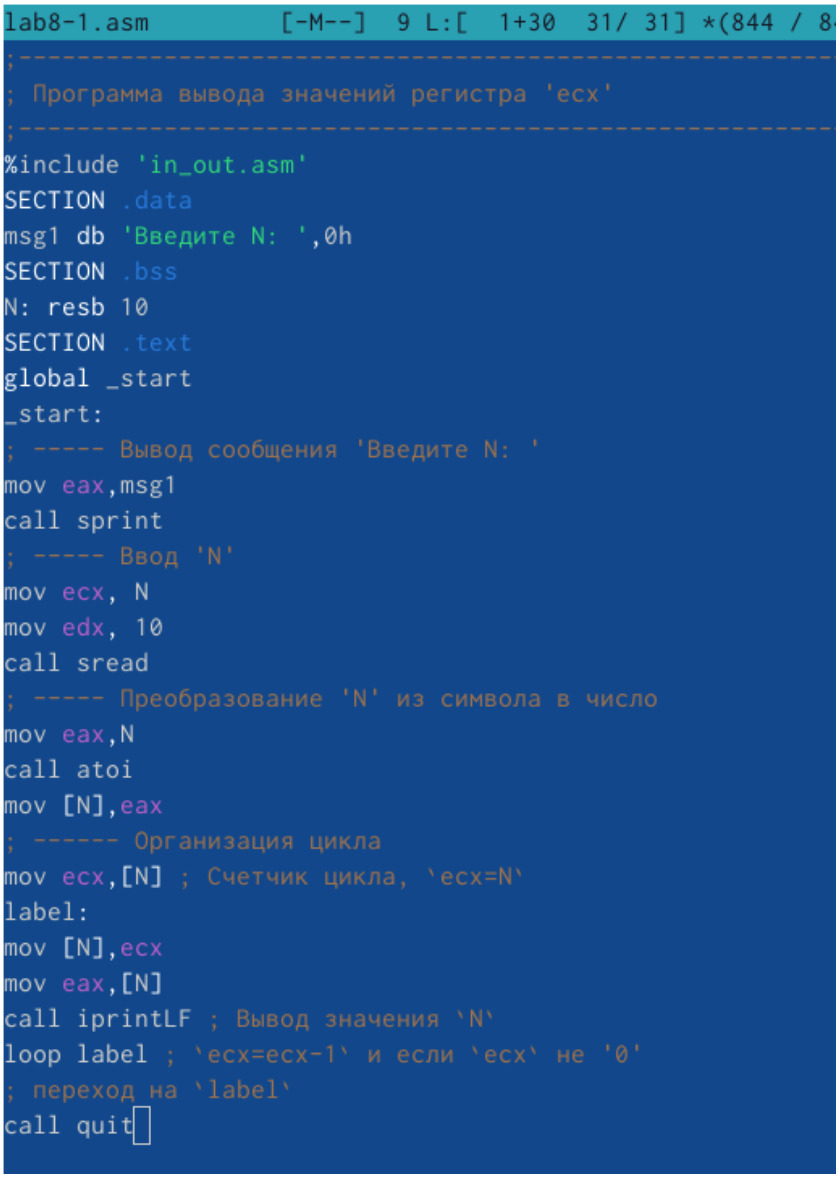

Рис. 4.2: Текст программы в файле lab8-1.asm

Создаем исполняемый файл и проверяем его работу (рис. 4.3).

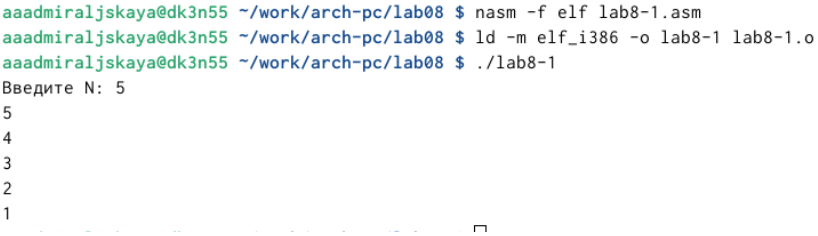

Рис. 4.3: Цикл, выводящий цифры от 5 до 1

Затем в этом же файле изменяем текст программы добавив изменение значение регистра ecx в цикле (рис. 4.4).

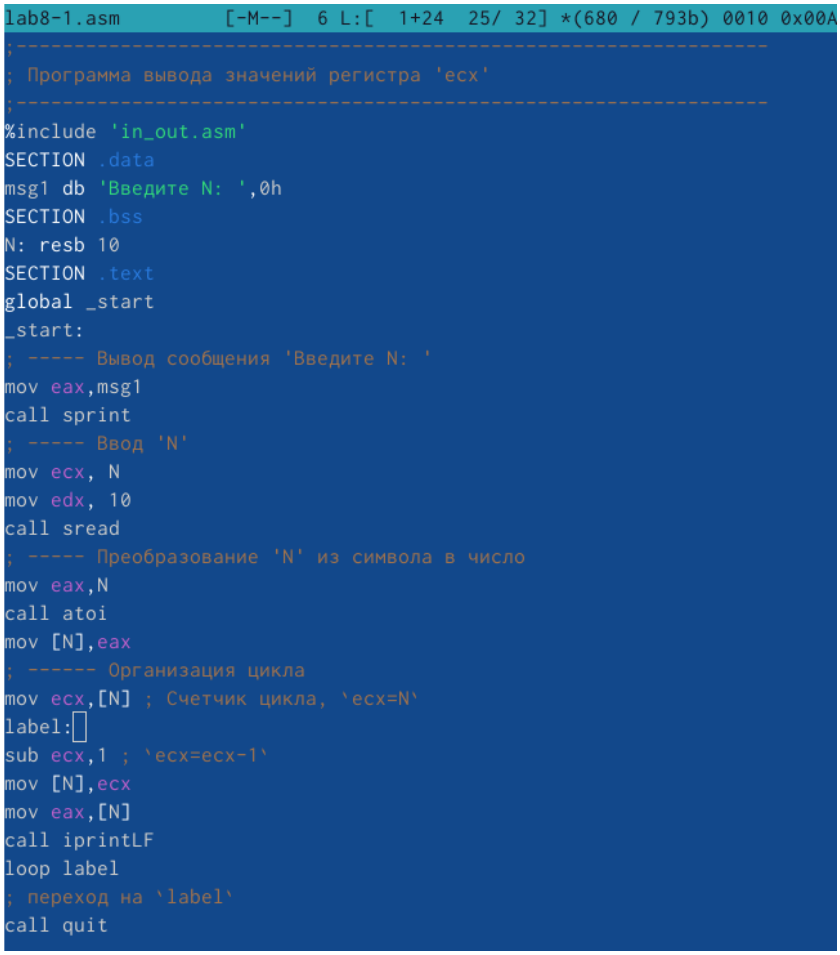

Рис. 4.4: Измененный текст программы

Создаем исполняемый файл и проверяем его работу (рис. 4.5).

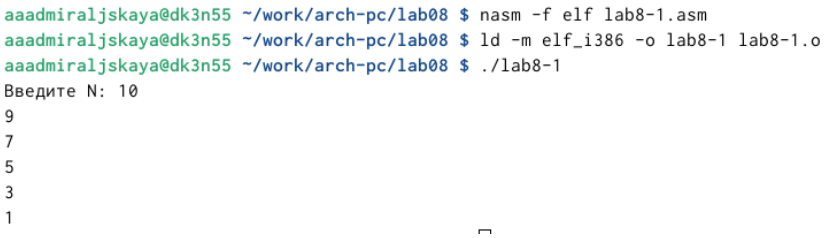

Рис. 4.5: Цикл, выводящий нечетные цифры от 10 до 1

Далее в этом же файле вносим изменения в текст программы добавив коман-

ды push и pop для сохранения значения счетчика цикла loop (рис. 4.6).

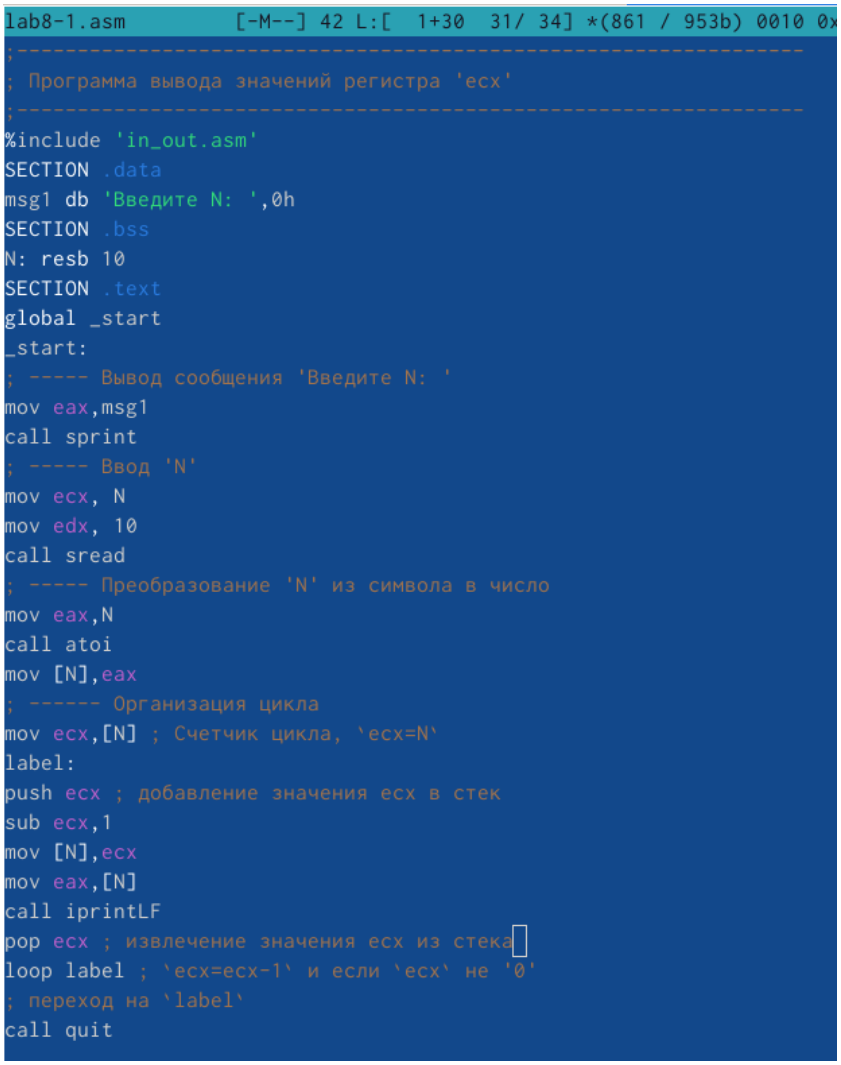

Рис. 4.6: Внесенные в файл изменения

Создаем исполняемый файл и проверяем его работу (рис. 4.7).

```
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Рис. 4.7: Цикл, выводящий цифры от 9 до 0

Следующим шагом создаем файл lab8-2.asm и вводим в него текст программы из листинга 8.2 (рис. 4.8).

```
lab8-2.asm          [-M--]  9 L:[  1+22  23/ 23] *(115
;-------------------------------------------------------
; 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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 4.8: Текст программы в файле lab8-2.asm

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. 4.9).

```
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-2 S a s h a
S
a
s
h
a
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $
```

Рис. 4.9: Работа файла с указанными аргументами

Теперь создаем файл lab8-3.asm и вводим в него текст программы из листинга 8.3 (рис. 4.10).

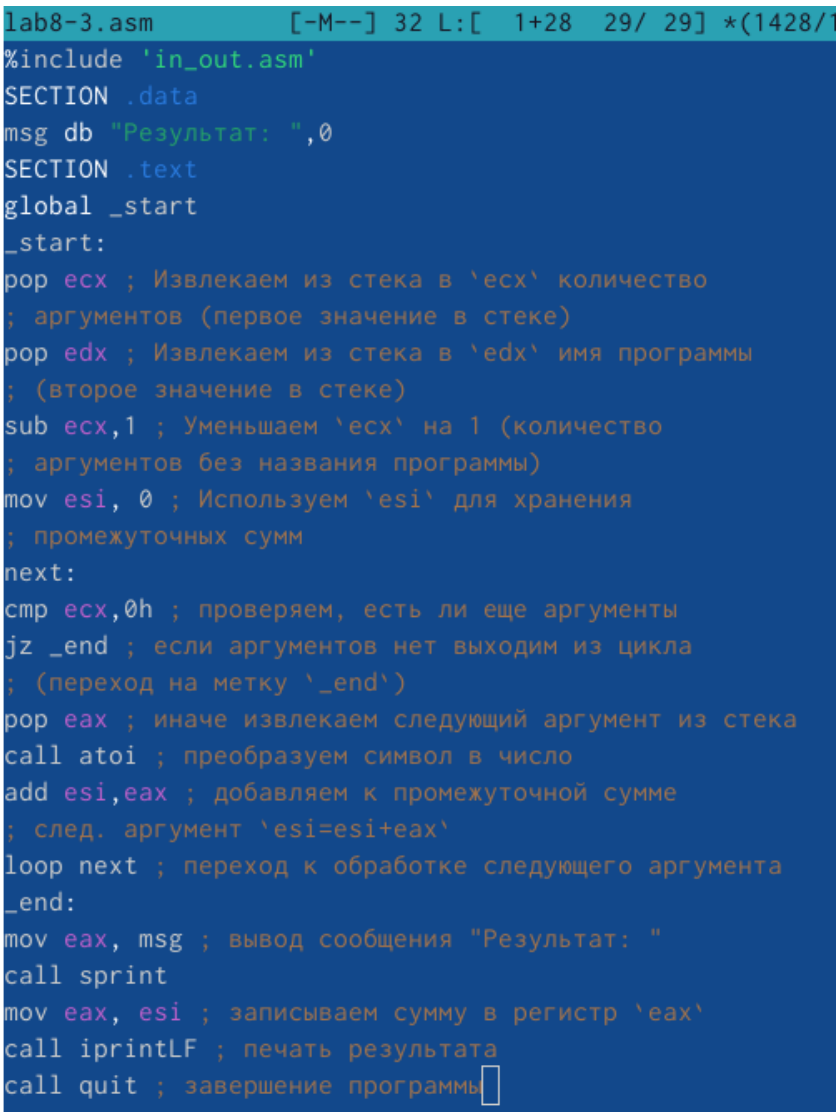

Рис. 4.10: Текст программы в файле lab8-3.asm

Создаем исполняемый файл и запускаем его, указав аргументы (рис. 4.11).

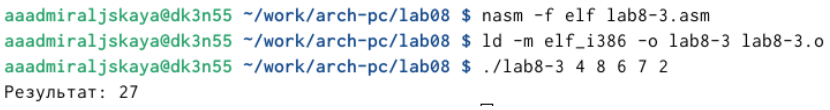

Рис. 4.11: Работа программы, которая складыает числа, введенные пользователем

В этом же файле lab8-3.asm изменяем программу так, чтобы она умножала введенные числа (рис. 4.12).

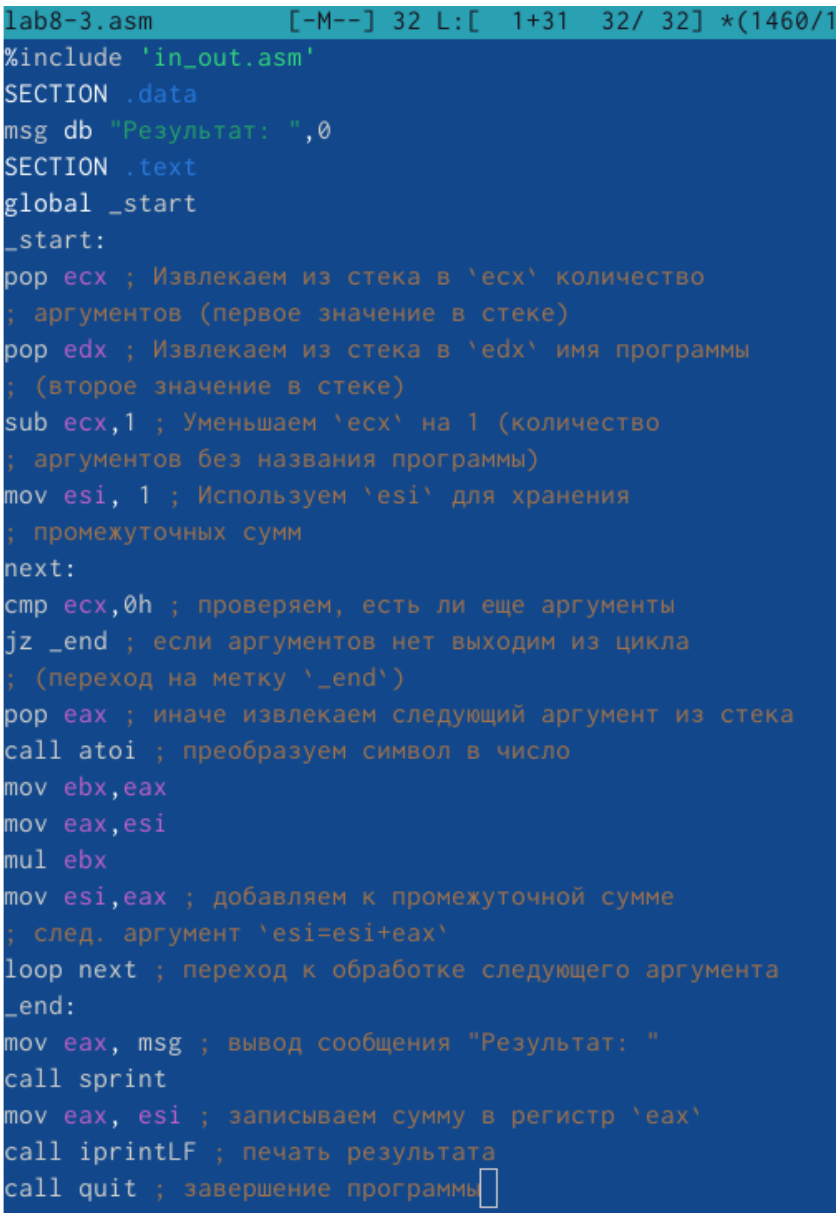

Рис. 4.12: Измененный текст файла lab8-3.asm

Создаем исполняемый файл и запускаем его, указав аргументы (рис. 4.13).

```
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $  nasm -f elf lab8-3.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-3 4 8 6 7 2
Результат: 2688
```

Рис. 4.13: Работа программы, которая умножает числа, введенные пользователем

Приступим к выполнению задания для самостоятельной работы. Создаем

файл lab8-4.asm и вводим в него программу, которая находит сумму значений функции. Вид функции берем из варианта №7 (рис. 4.14).

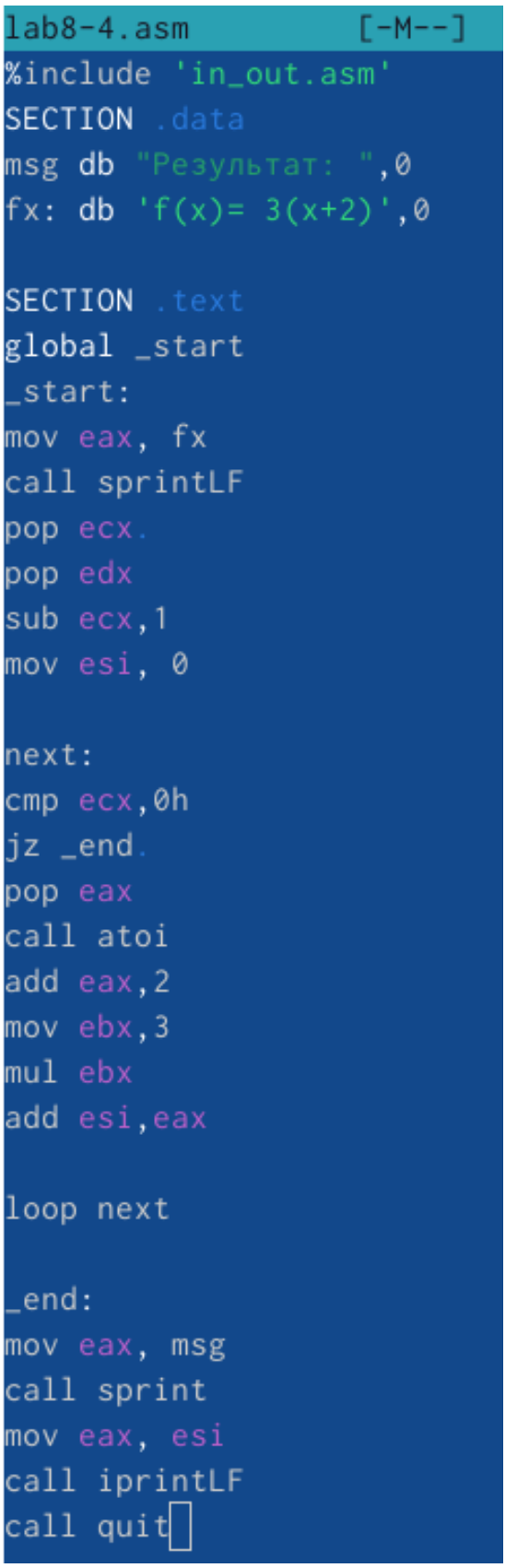

Рис. 4.14: Программа, находящая сумму значений функции

Создаем исполняемый файл и проверяем его работу для значений: x=1, x=2, x=3, x=4 - указываем цифры как аргументы (рис. 4.15).

```
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-4 1
f(x)= 3(x+2)
Результат: 9
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-4 2
f(x)= 3(x+2)
Результат: 12
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-4 3
f(x)= 3(x+2)
Результат: 15
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-4 4
f(x)= 3(x+2)
Результат: 18
```

Рис. 4.15: Работы программы при разных значениях x

Листинг программы из задания самостоятельной работы:

%include ‘in_out.asm’ SECTION .data msg db “Результат:”,0 fx: db ‘f(x)= 3(x+2)’,0

SECTION .text global _start _start: mov eax, fx call sprintLF pop ecx pop edx sub ecx,1 mov esi, 0

next: cmp ecx,0h jz _end pop eax call atoi add eax,2 mov ebx,3 mul ebx add esi,eax loop next

_end: mov eax, msg call sprint mov eax, esi call iprintLF call quit

5 Выводы

В процессе выполнения лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O’Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O’Reilly Media, 2005. 354 с.