Отчёт по лабораторной работе №8

Иванова Анастасия Сергеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобрети навыки написания программ с использованием циклов и обработкой аргументов командной строки.

2 Порядок выполнения лабораторной работы

1. Создадим каталог для программам лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm рис. 2.1:

Рисунок 2.1: Создание каталога и файла

Введем в файл lab8-1.asm текст программы, создадим исполняемый файл и проверим его работу рис. 2.2:

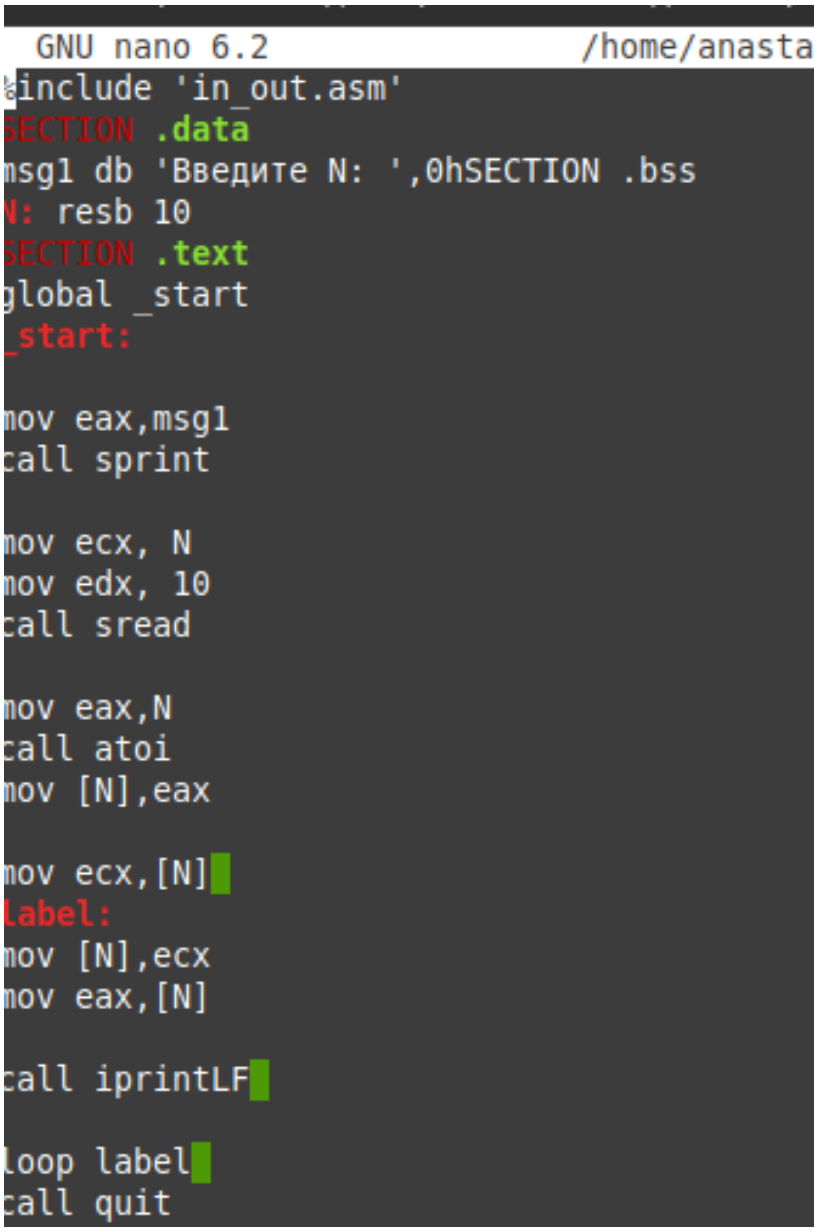

Рисунок 2.2: Ввод программы

рис. 2.3:

Рисунок 2.3: Проверка работы программы

рис. 2.4:

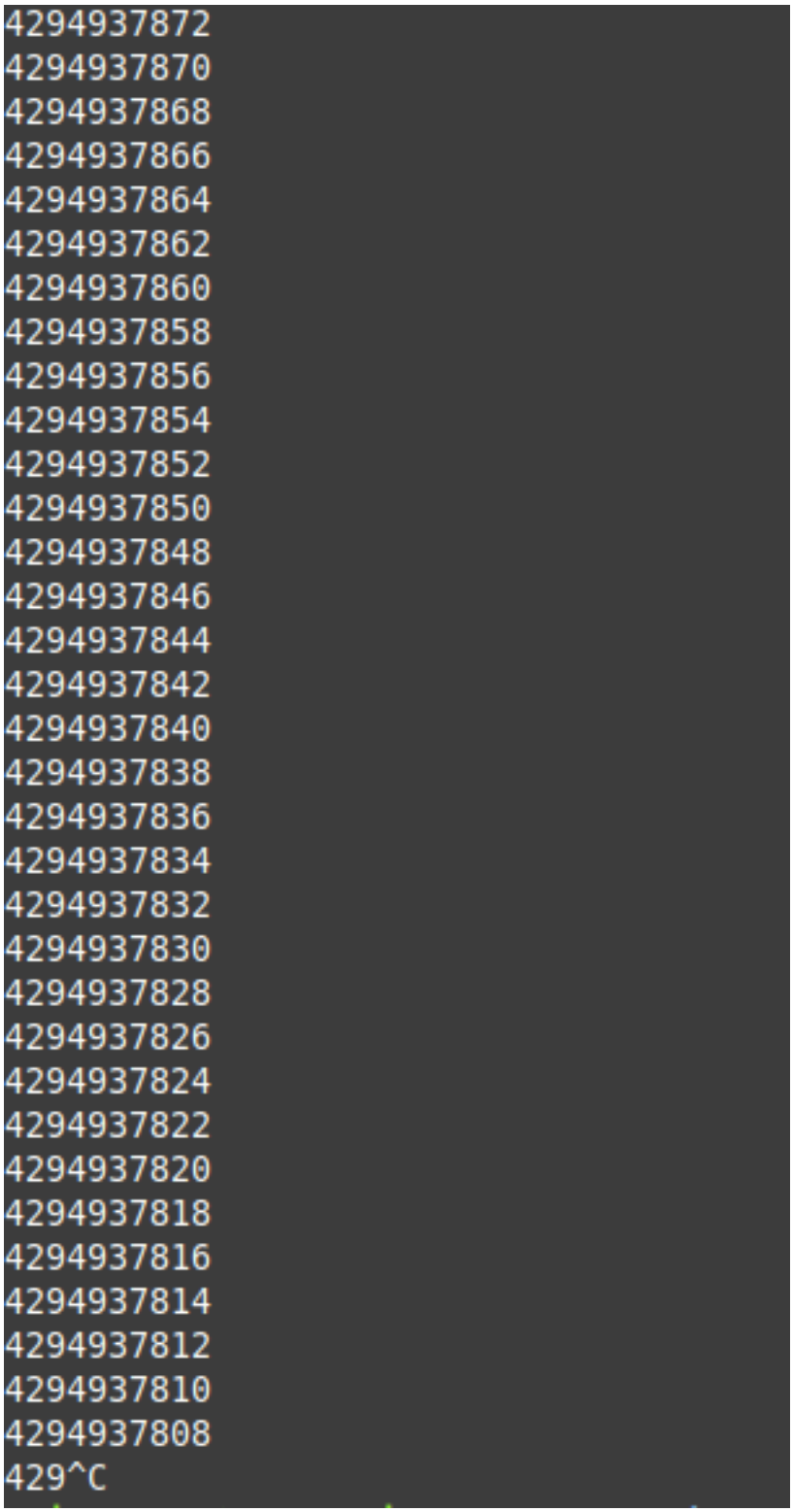

Рисунок 2.4: Бесконечный вывод

Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы, при вводе некоторых значений выводит бесконечный цикл.

Изменим текст программы рис. 2.5:

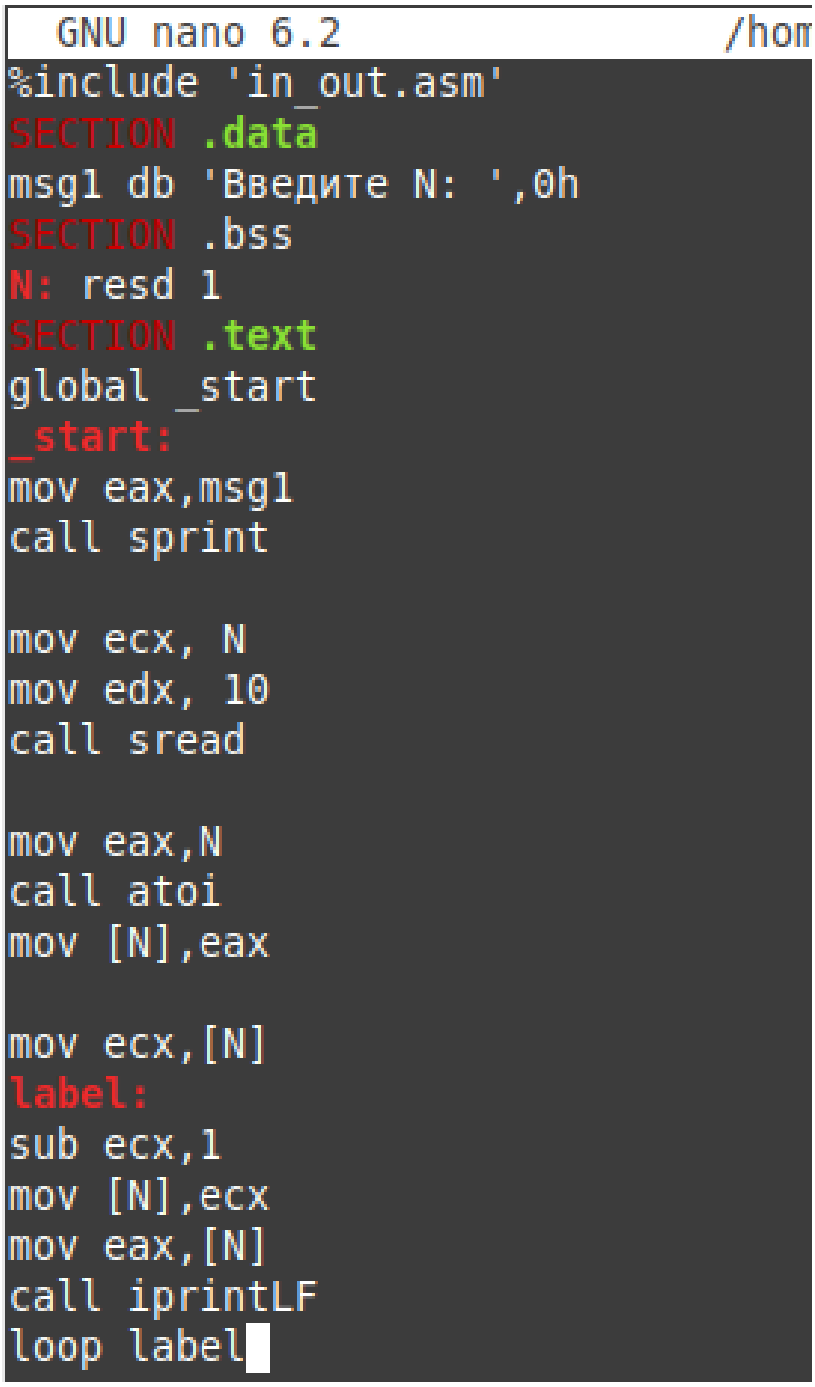

Рисунок 2.5: Изменение программы

Создадим исполняемый файл и проверим его работу рис. 2.6:

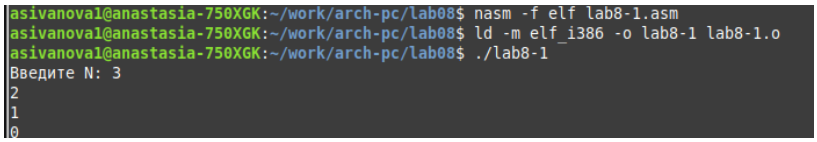

Рисунок 2.6: Проверка работы программы

Какие значения принимает регистр ecx в цикле?: 3 ⟶ 3 ⟶ 2 ⟶ 3 ⟶ 2 ⟶ 2 ⟶ 1 ⟶ 2 ⟶ 1 ⟶ 1 ⟶ 0 ⟶ 1 ⟶ 0 (цикл выполняется ровно 3 раза) Соответствует ли число проходов цикла значению введенному с клавиатуры?: да

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесем изменения в текст программы добавив команды push и pop рис. 2.7:

```
 GNU nano 6.2                    /home/ana
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax
mov ecx,[N]

label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
```

Рисунок 2.7: Изменение программы

Создадим исполняемый файл и проверим его работу рис. 2.8:

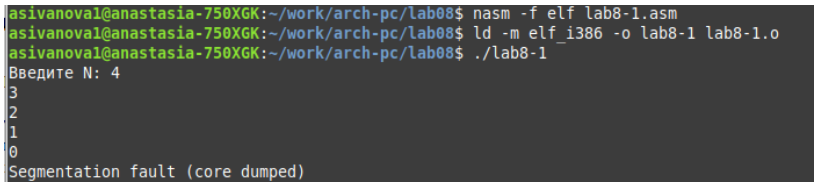

Рисунок 2.8: Проверка работы программы

2. При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки.

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы рис. 2.9:

Рисунок 2.9: Создание файла

рис. 2.10:

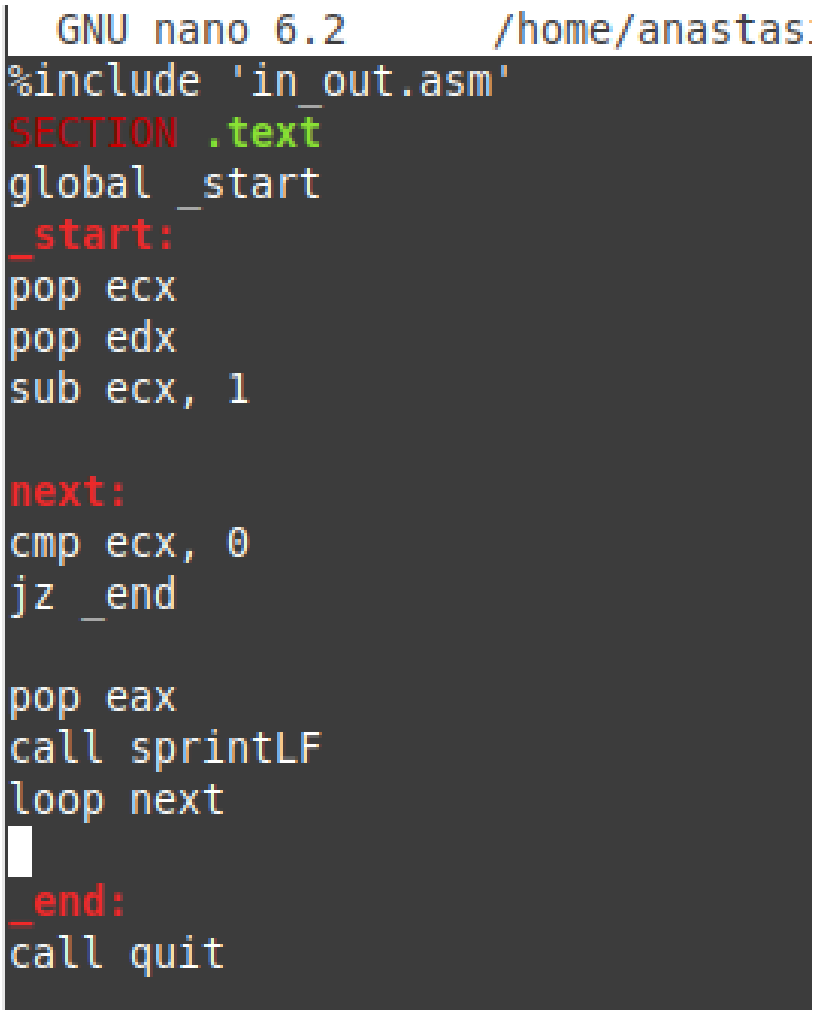

Рисунок 2.10: Ввод программы

Создадим исполняемый файл и запустим его, указав аргументы: user@dk4n31:~$ ./lab8-2 аргумент1 аргумент 2 „аргумент 3“ рис. 2.11:

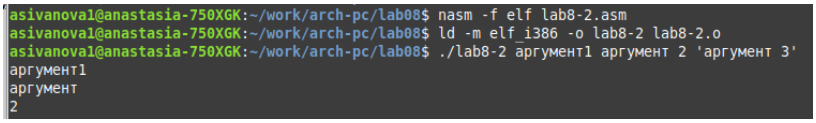

Рисунок 2.11: Проверка работы программы

Программа обработала 4 аргумента.

3. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.

Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы рис. 2.12:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ touch lab8-3.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ gedit lab8-3.asm
```

Рисунок 2.12: Создание файла

рис. 2.13:

```
  GNU nano 6.2
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
next:
cmp ecx,0h
jz _end
pop eax
call atoi
add esi,eax
loop next
_end:
mov eax, msg
call sprint
```

Рисунок 2.13: Ввод программы

Создадим исполняемый файл и запустим его, указав аргументы рис. 2.14:

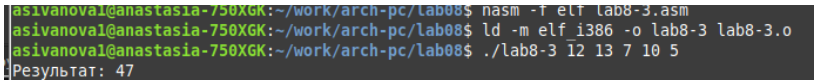

Рисунок 2.14: Проверка работы программы

Изменим текст программы для вычисления произведения аргументов командной строки рис. 2.15:

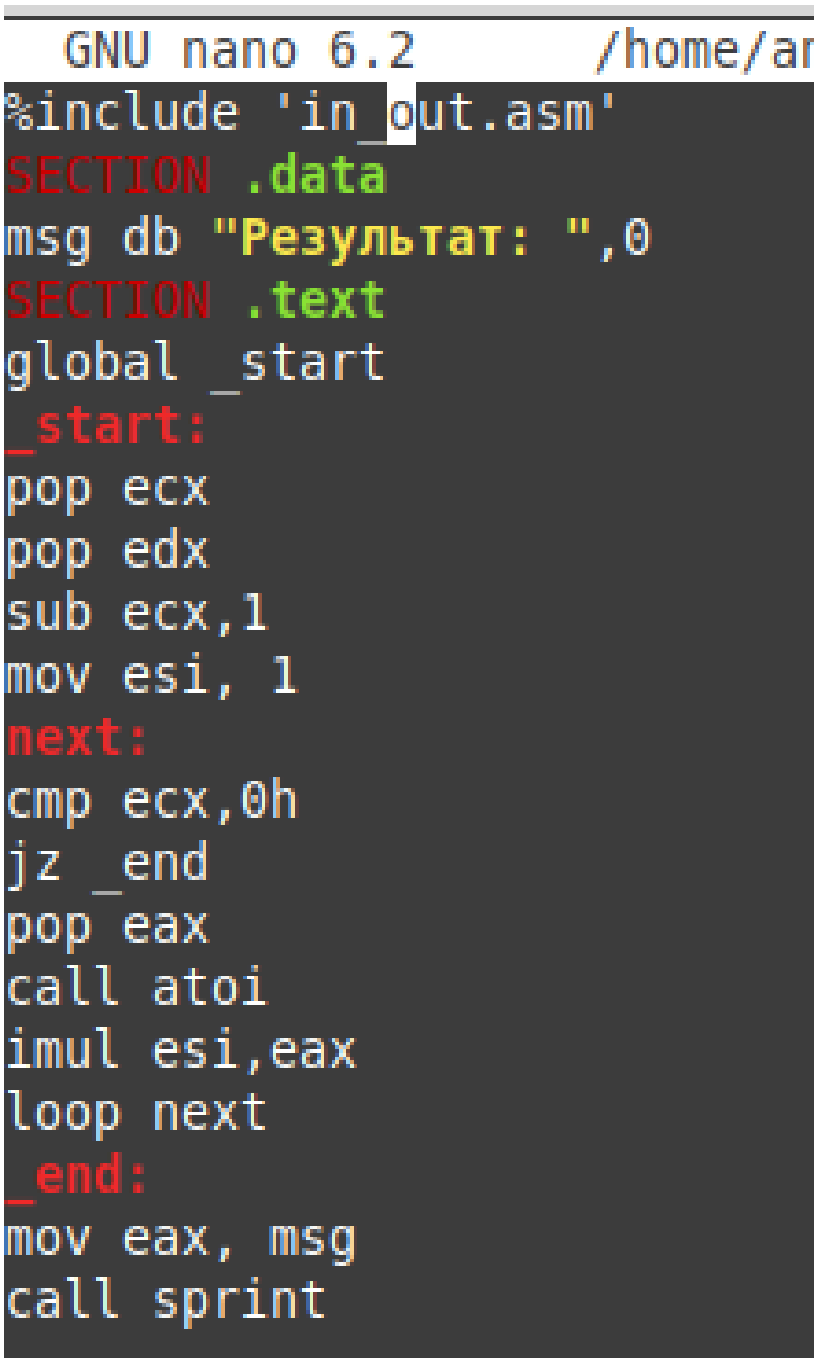

Рисунок 2.15: Изменение программы

Создадим исполняемый файл и запустим его, указав аргументы рис. 2.16:

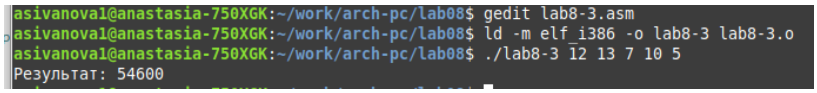

Рисунок 2.16: Проверка работы программы

3 Задание для самостоятельной работы

1. Создадим файл и введем в него программу рис. 3.1:

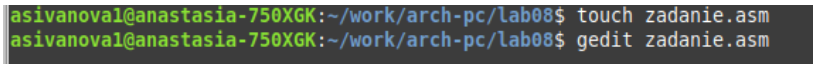

Рисунок 3.1: Создание файла

рис. 3.2:

```
  GNU nano 6.2
%include 'in_out.asm'
SECTION .data
func_msg db "Функция: f(x)=30x-11",0
result_msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0
jz _end

pop eax
call atoi

mov ebx, 30
mul ebx
sub eax, 11

add esi, eax

loop next

_end:
mov eax, func_msg
call sprintLF

mov eax, result_msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рисунок 3.2: Ввод программы

2. Создадим исполняемый файл и проверим работу программы рис. 3.3:

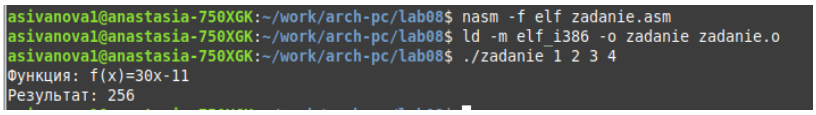

Рисунок 3.3: Проверка работы программы

4 Вывод

Мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.