

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Седохин Даниил Алексеевич

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы	17
4	Выводы	20

Список иллюстраций

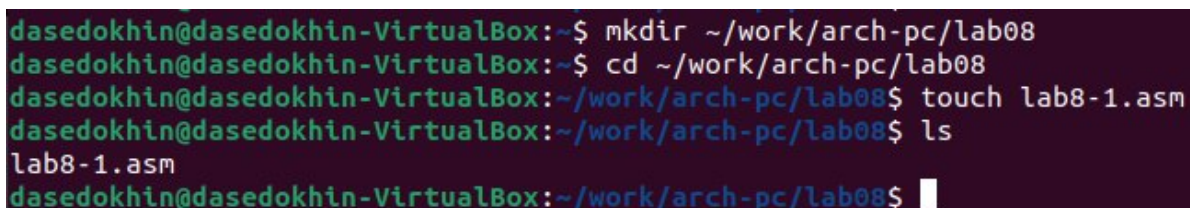
2.1	Создание каталога для лабораторной работы №8 и файла lab8-1.asm	5
2.2	Ввод в файл lab8-1.asm текст программы из листинга 8.1	6
2.3	Создание и проверка исполняемого файла lab8-1.asm	7
2.4	Изменение текста программы, изменив значение регистра esx . .	8
2.5	Создание и проверка исполняемого файла	9
2.6	Изменение текста программы, добавив команды push и pop . . .	10
2.7	Создание и проверка исполняемого файла	11
2.8	Создание файла lab8-2.asm	12
2.9	Ввод текст программы в файл lab7-2.asm из листинга 8.2.	12
2.10	Создание исполняемого файла и его проверка с указанием аргу- ментов	13
2.11	Создание файла lab8-3.asm в каталоге ~/work/arch-pc/lab08	13
2.12	Ввод текста программы в файл lab8-3.asm из листинга 8.3.	14
2.13	Создание и проверка исполняемого файла lab8-3.asm	14
2.14	Изменение текста программы lab8-3.asm для вычисления произве- дения аргументов командной строки	15
2.15	Создание и проверка исполняемого изменённого файла lab8-3.asm	16
3.1	Создание файла srab1.asm	17
3.2	Пишем код программы для 6-го варианта	18
3.3	Создание и проверка исполняемого файла	19

1 Цель работы

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

2 Выполнение лабораторной работы

- 1) Создадим каталог для программ лабораторной работы № 8, перейдём в него и создадим файл lab8-1.asm: (рис. 2.1).



```
dasedokhin@dasedokhin-VirtualBox:~$ mkdir ~/work/arch-pc/lab08
dasedokhin@dasedokhin-VirtualBox:~$ cd ~/work/arch-pc/lab08
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ touch lab8-1.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ls
lab8-1.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.1: Создание каталога для лабораторной работы №8 и файла lab8-1.asm

- 2) При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра ecx. Введём в файл lab8-1.asm текст программы из листинга 8.1. (рис. 2.2).

```
dasedokhin@dasedokhin-VirtualBox: ~/work/arch-pc/...
GNU nano 6.2 /home/dasedokhin/work/arch-pc/lab08/lab8-1.asm *
;-----
; Программа вывода значений регистра 'ecx'
;-----
#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:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.2: Ввод в файл lab8-1.asm текст программы из листинга 8.1

Создадим исполняемый файл и проверим его работу. (рис. 2.3).

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

Рис. 2.3: Создание и проверка исполняемого файла lab8-1.asm

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

Изменим текст программы добавив изменение значение регистра `ecx` в цикле: (рис. 2.4).

```
dasedokhin@dasedokhin-VirtualBox: ~/work/arch-pc/...
GNU nano 6.2 /home/dasedokhin/work/arch-pc/lab08/lab8-1.asm *
;-----
; Программа вывода значений регистра 'ecx'
;-----
#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:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.4: Изменение текста программы, изменив значение регистра ecx

Создадим исполняемый файл и проверим его работу. (рис. 2.5).


```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.5: Создание и проверка исполняемого файла

Число проходов цикла не соответствуют введённому значению с клавиатуры. Программа обрабатывает только нечетные числа.

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек.

Внесём изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop` (рис. 2.6).

```
dasedokhin@dasedokhin-VirtualBox: ~/work/arch-pc/...
GNU nano 6.2 /home/dasedokhin/work/arch-pc/lab08/lab8-1.asm *
; -----
; Программа вывода значений регистра 'ecx'
; -----
#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 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
pop ecx ; извлечение значения ecx из стека
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.6: Изменение текста программы, добавив команды push и pop

Создадим исполняемый файл и проверим его. (рис. 2.7).

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

Рис. 2.7: Создание и проверка исполняемого файла

В данном случае число переходов соответствует числу введённому с клавиатуры.

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обратном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. 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 (рис. 2.8).

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ touch lab8-2.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.8: Создание файла lab8-2.asm

Введём в него текст программы из листинга 8.2. (рис. 2.9).

```
GNU nano 6.2 /home/dasedokhin/work/arch-pc/lab08/lab8-2.asm *
;-----
; Обработка аргументов командной строки
;-----
#include 'in_out.asm'
SECTION .text
global _start
_start:
пор есх ; Извлекаем из стека в `есх` количество
; аргументов (первое значение в стеке)
пор edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub есх, 1 ; Уменьшаем `есх` на 1 (количество
; аргументов без названия программы)
next:
cmp есх, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
пор еах ; 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. 2.9: Ввод текст программы в файл lab7-2.asm из листинга 8.2.

Создание исполняемого файла и его запуск, с указанием аргументов. (рис. 2.10).

```

dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.
asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o l
ab8-2 lab8-2.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1
аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 2.10: Создание исполняемого файла и его проверка с указанием аргументов

Программой было обработаны все 3 аргумента

Рассмотрим еще один пример программы которая выводит сумму чисел, которые пере- даются в программу как аргументы. Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. 2.11).

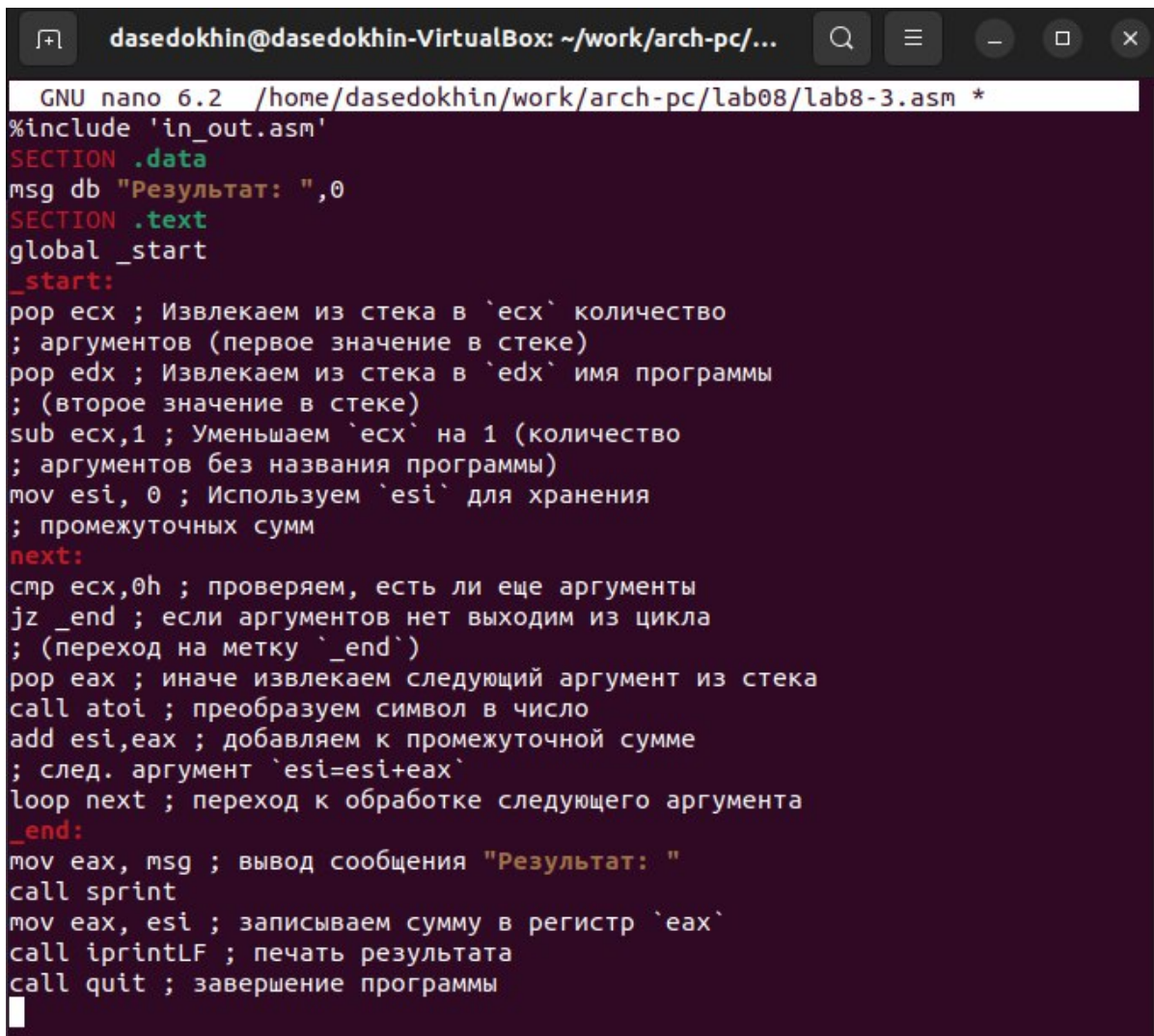
```

dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ touch lab8-3.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.o
lab8-1      lab8-1.o    lab8-2.asm  lab8-3.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 2.11: Создание файла lab8-3.asm в каталоге ~/work/arch-pc/lab08

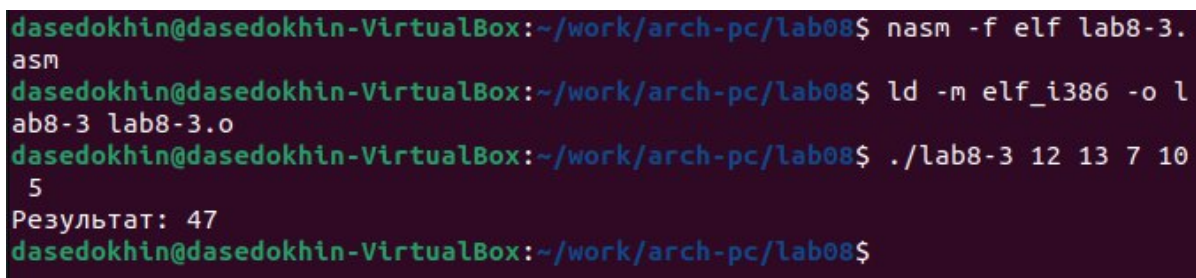
Введём в него текст программы из листинга 8.3. (рис. 2.12).



```
GNU nano 6.2 /home/dasedokhin/work/arch-pc/lab08/lab8-3.asm *
%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, 0 ; 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
add 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
```

Рис. 2.12: Ввод текста программы в файл lab8-3.asm из листинга 8.3.

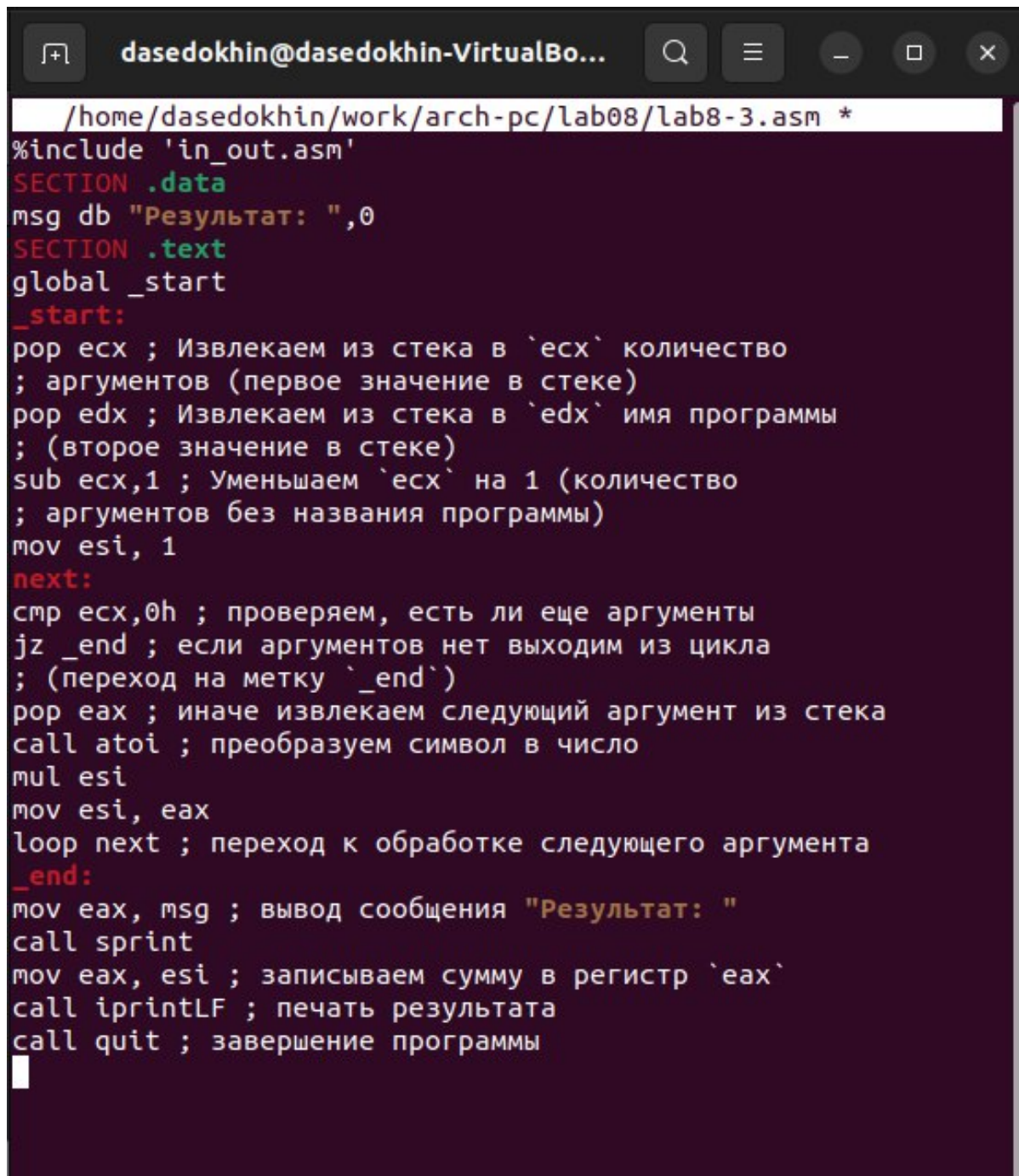
Создадим исполняемый файл и запустим его, указав аргументы. (рис. 2.13).



```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.
asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o l
ab8-3 lab8-3.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10
5
Результат: 47
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.13: Создание и проверка исполняемого файла lab8-3.asm

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. 2.14).



```
/home/dasedokhin/work/arch-pc/lab08/lab8-3.asm *
#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
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
    mul esi
    mov 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
```

Рис. 2.14: Изменение текста программы lab8-3.asm для вычисления произведения аргументов командной строки

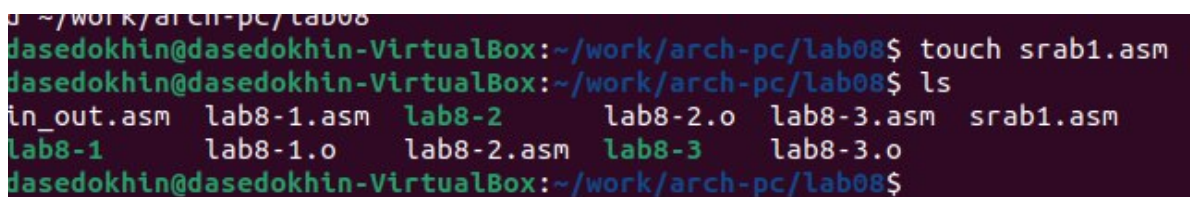
Создадим исполняемый файл и проверим его, указав аргументы. (рис. 2.15).

```
Результат: 34000
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ nasm
-f elf lab8-3.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ld -m
elf_i386 -o lab8-3 lab8-3.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ./lab
8-3 1 5 10
Результат: 50
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.15: Создание и проверка исполняемого изменённого файла lab8-3.asm

3 Задание для самостоятельной работы

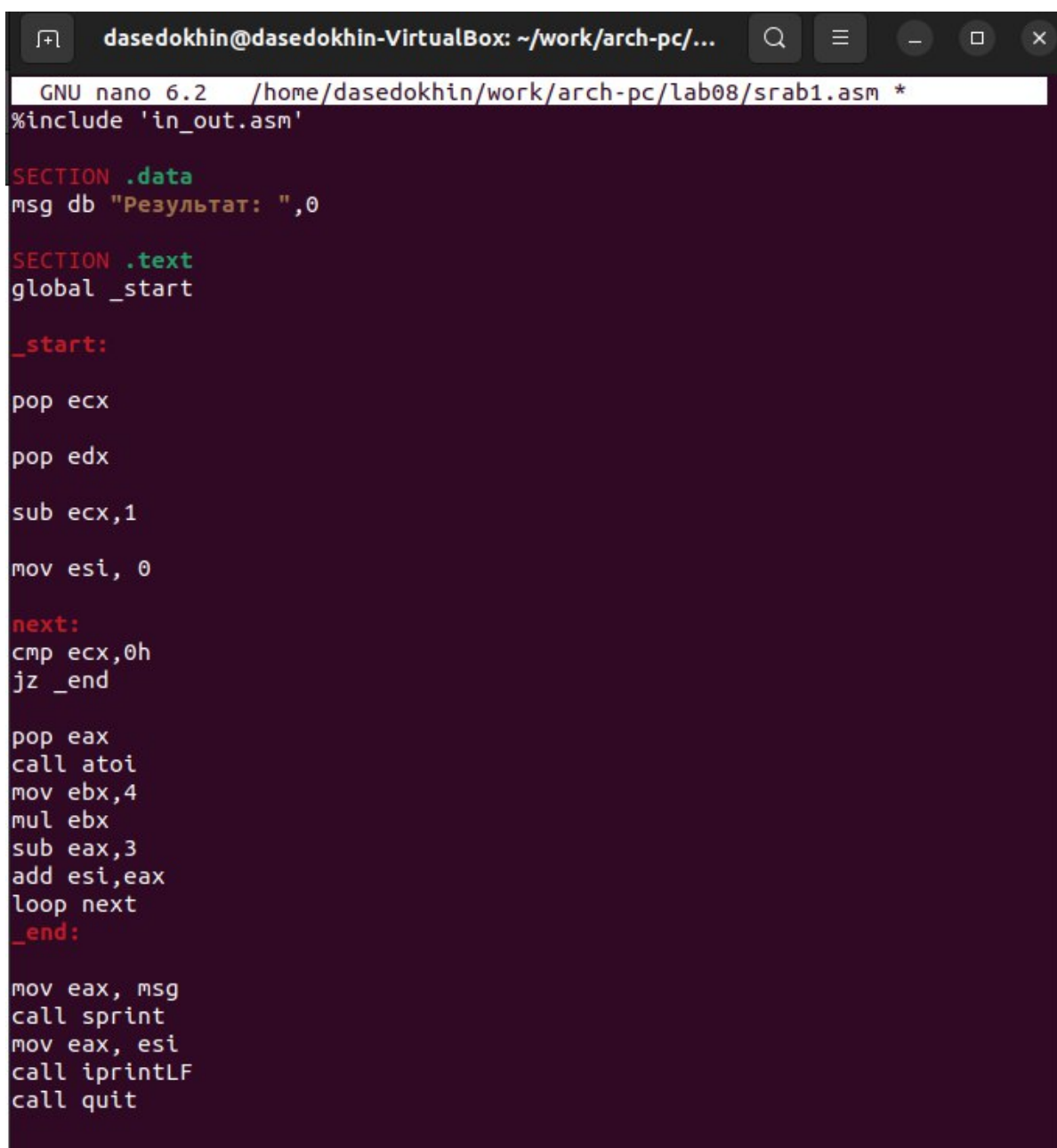
Создадим файл `srab1.asm` в каталоге `~/work/arch-pc/lab08` (рис. 3.1).

A screenshot of a terminal window with a dark background and light-colored text. The prompt is `dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$`. The first command is `touch srab1.asm`. The second command is `ls`, which lists the contents of the directory: `in_out.asm lab8-1.asm lab8-2 lab8-2.o lab8-3.asm srab1.asm lab8-1 lab8-1.o lab8-2.asm lab8-3 lab8-3.o`. The prompt returns to `dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$`.

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ touch srab1.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ls
in_out.asm lab8-1.asm lab8-2 lab8-2.o lab8-3.asm srab1.asm
lab8-1 lab8-1.o lab8-2.asm lab8-3 lab8-3.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 3.1: Создание файла `srab1.asm`

Напишем 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_i передаются как аргументы. Вид функции $f(x)$ возьмём из таблицы 8.1 вариантов заданий в соответствии с 6 вариантом. Моя функция будет следующей: $f(x) = 4x - 3$ (рис. 3.2).



```
dasedokhin@dasedokhin-VirtualBox: ~/work/arch-pc/...
GNU nano 6.2 /home/dasedokhin/work/arch-pc/lab08/srab1.asm *
#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
mov ebx,4
mul ebx
sub eax,3
add esi,eax
loop next
_end:

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

Рис. 3.2: Пишем код программы для 6-го варианта

Создадим исполняемый файл и проверим его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$ (рис. 3.3).

```
[0] + Обновлен файл
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf srab1.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o srab1 srab1.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$ ./srab1 1 2 3 4
Результат: 28
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 3.3: Создание и проверка исполняемого файла

4 Выводы

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