

Отчёт по лабораторной работе 8

Архитектура компьютеров

Эйвази Мани НПИбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация циклов в NASM	6
2.2	Самостоятельное задание	16
3	Выводы	19

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	15
2.14	Программа lab8-prog.asm	17
2.15	Запуск программы lab8-prog.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).



Рис. 2.1: Создан каталог

При реализации циклов в NASM с использованием инструкции `loop` необходимо помнить, 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`.

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

```
manieyvazi@VirtualBox: ~/work/arch-pc/lab0
GNU nano 4.8 lab8-1.asm
#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: Программа lab8-1.asm

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

Рис. 2.3: Запуск программы lab8-1.asm

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

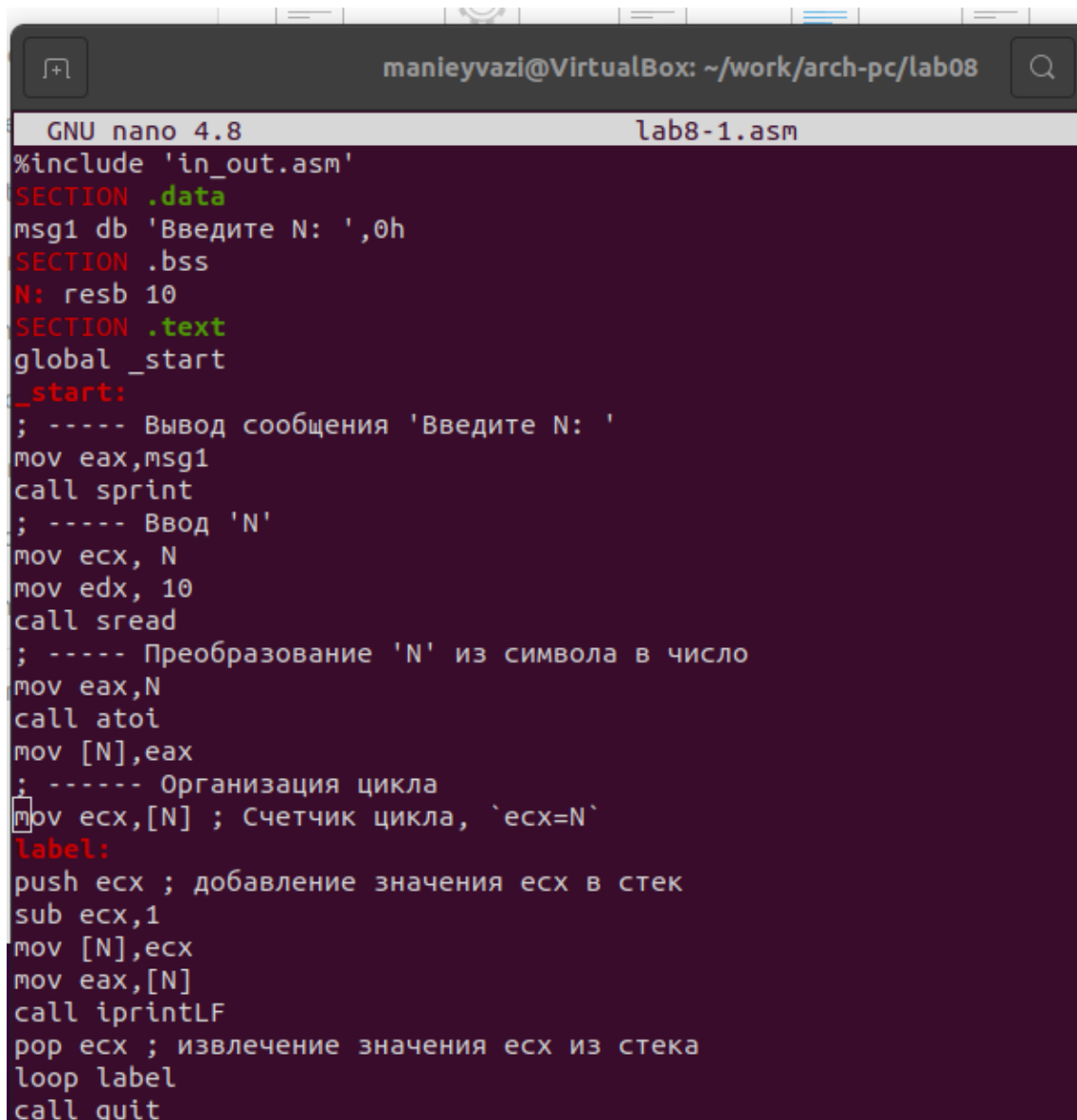

```
manieyvazi@VirtualBox: ~/work/arc
GNU nano 4.8 lab8-1.asm
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- 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
loop label
; переход на `label`
call quit
```

Рис. 2.4: Программа lab8-1.asm

```
4294935350
4294935358
4294935356
4294935354
4294935352
4294935350
4294935348
42^C
manleyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
manleyvazi@VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.5: Запуск программы lab8-1.asm

Для корректного использования регистра `ecx` в цикле можно использовать стек. Внес изменения в текст программы, добавив команды `push` и `pop` для сохранения и восстановления значения счетчика цикла `loop` (рис. 2.6). Создал исполняемый файл и проверил его работу (рис. 2.7). Программа корректно выводит числа от $N-1$ до 0, при этом число проходов цикла соответствует N .



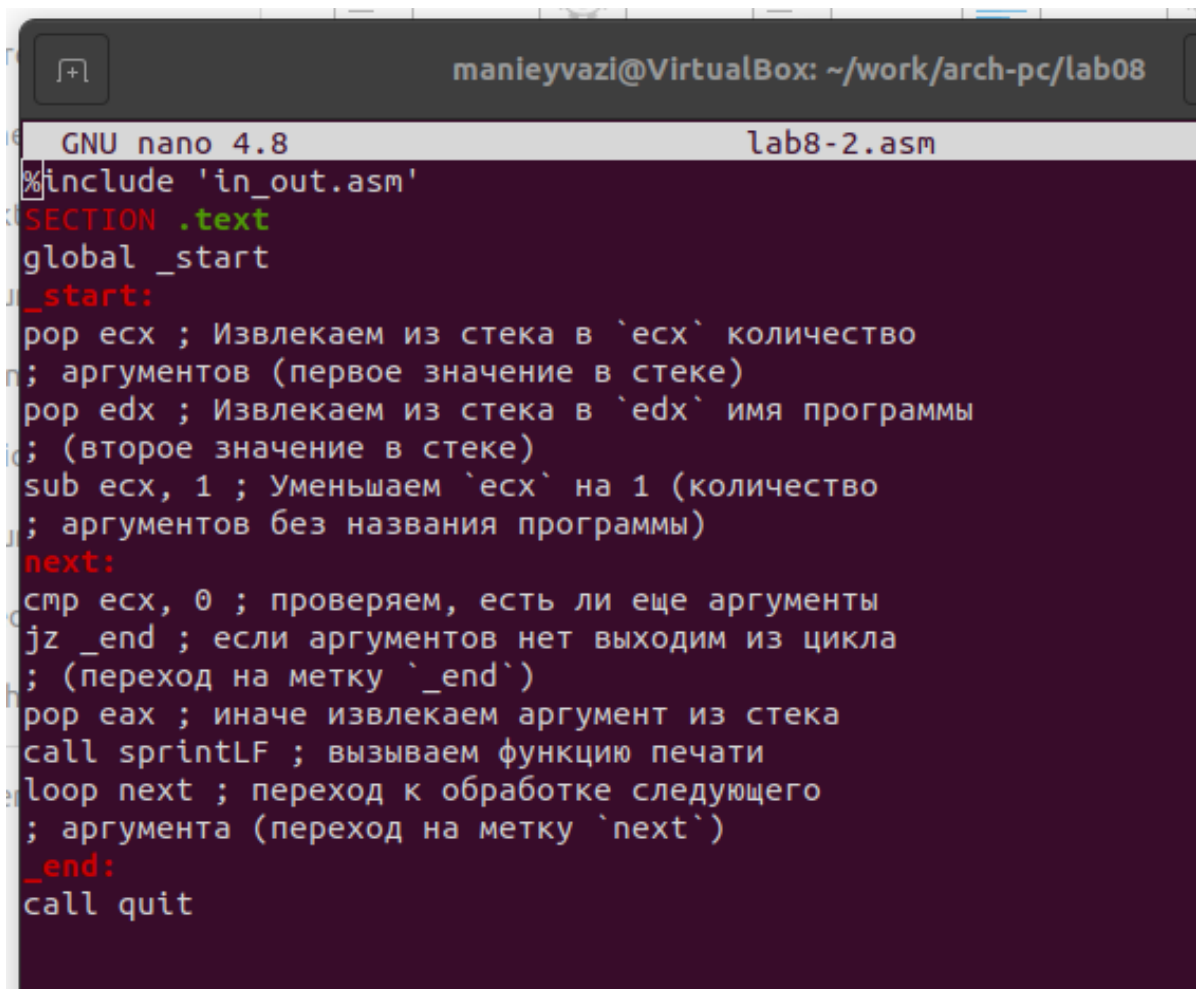
```
manieyvazi@VirtualBox: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-1.asm
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call 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
```

Рис. 2.6: Программа lab8-1.asm

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

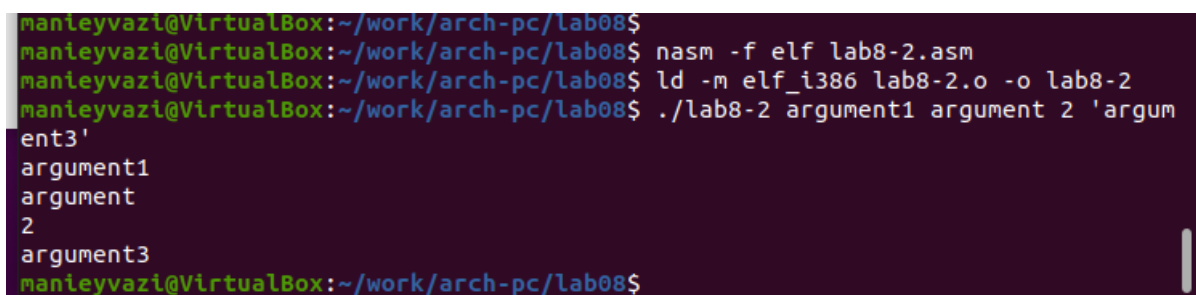
Рис. 2.7: Запуск программы lab8-1.asm

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



```
manieyvazi@VirtualBox: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-2.asm
%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 sprintf ; вызываем функцию печати
    loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
    call quit
```

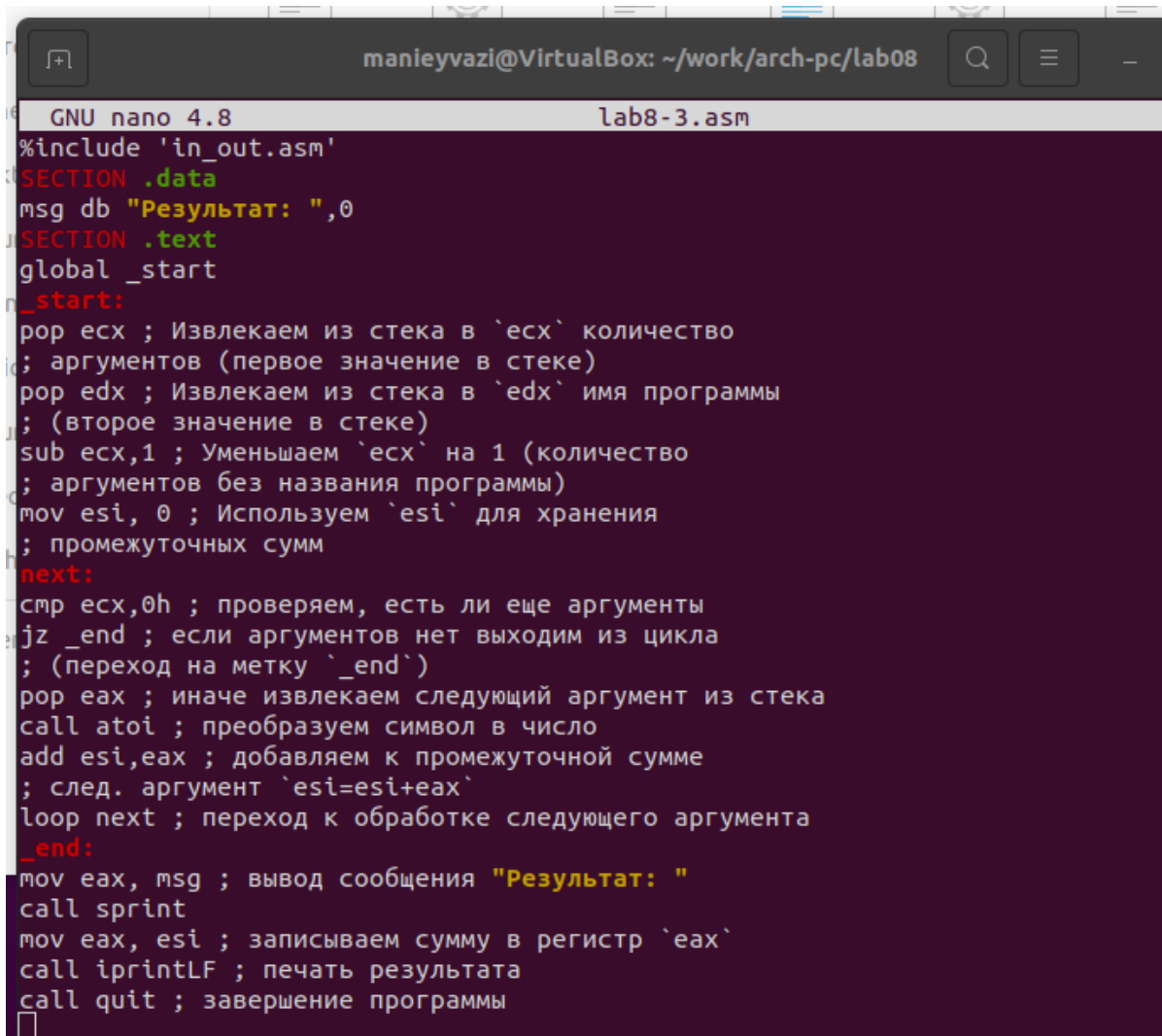
Рис. 2.8: Программа lab8-2.asm



```
manieyvazi@VirtualBox:~/work/arch-pc/lab08$
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 argument1 argument 2 'argum
ent3'
argument1
argument
2
argument3
manieyvazi@VirtualBox:~/work/arch-pc/lab08$
```

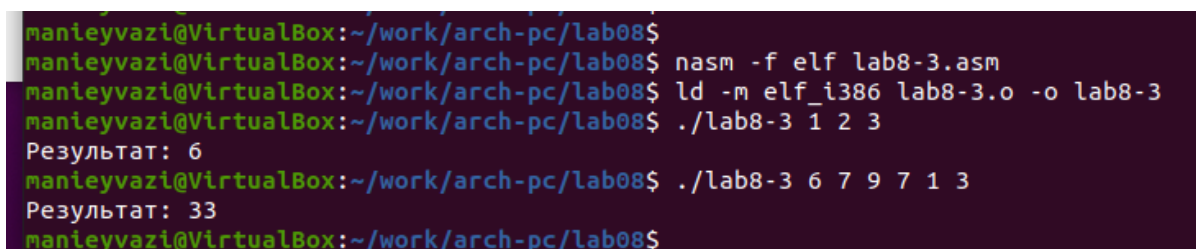
Рис. 2.9: Запуск программы lab8-2.asm

Рассмотрел пример программы, которая вычисляет сумму чисел, переданных в программу в качестве аргументов (рис. 2.10, рис. 2.11).



```
manieyvazi@VirtualBox: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-3.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por 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`)
por 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
```

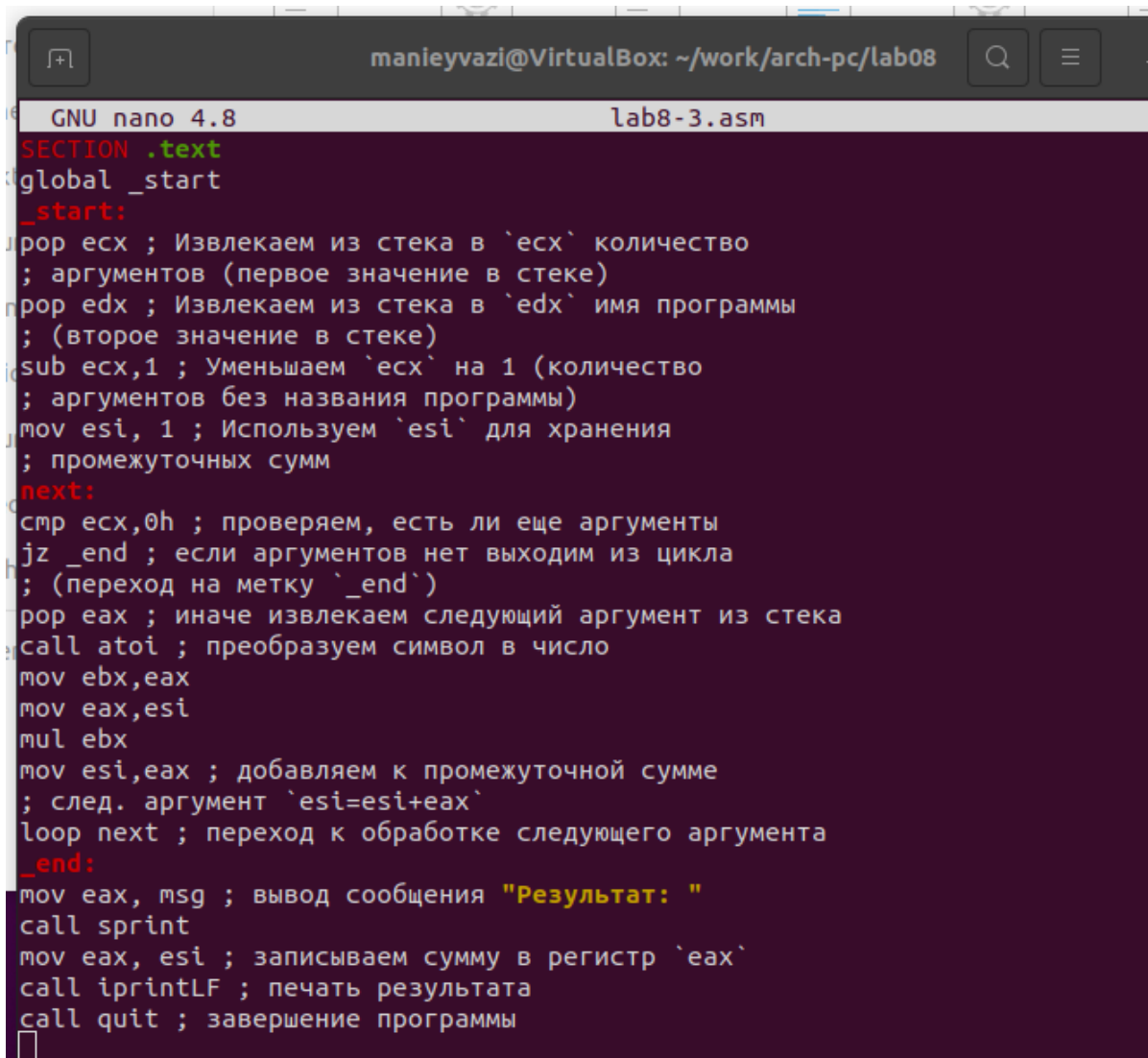
Рис. 2.10: Программа lab8-3.asm



```
manieyvazi@VirtualBox:~/work/arch-pc/lab08$
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 1 2 3
Результат: 6
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 6 7 9 7 1 3
Результат: 33
manieyvazi@VirtualBox:~/work/arch-pc/lab08$
```

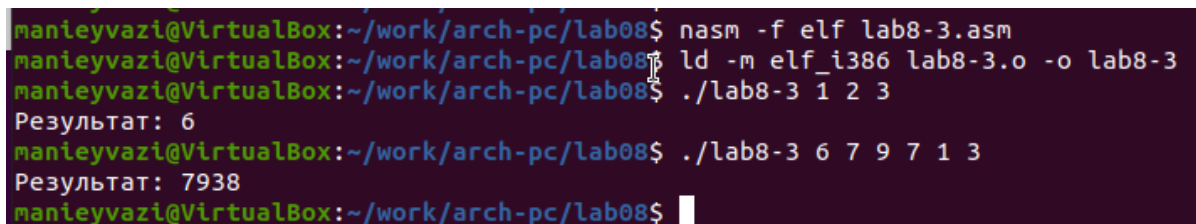
Рис. 2.11: Запуск программы lab8-3.asm

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



```
manieyvazi@VirtualBox: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-3.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)
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 iprintf ; печать результата
call quit ; завершение программы
```

Рис. 2.12: Программа lab8-3.asm



```
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 1 2 3
Результат: 6
manieyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 6 7 9 7 1 3
Результат: 7938
manieyvazi@VirtualBox:~/work/arch-pc/lab08$
```

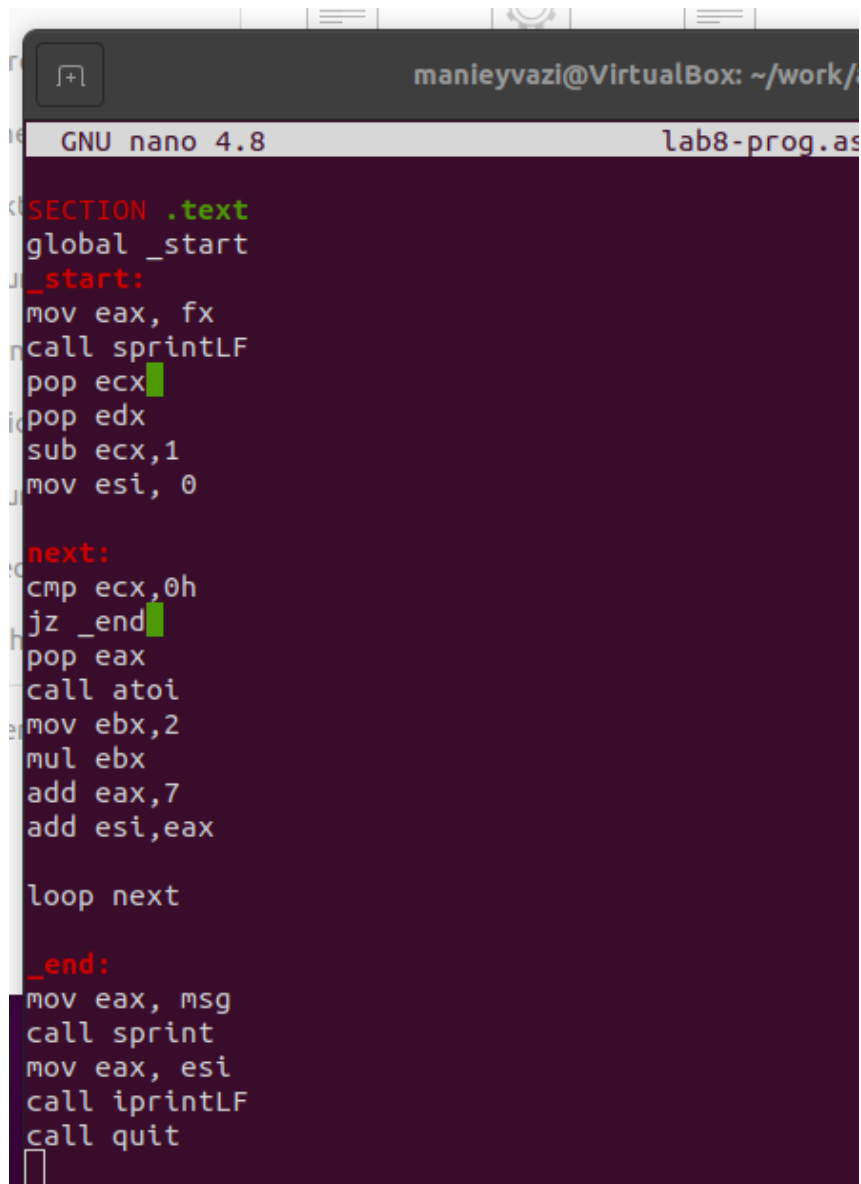
Рис. 2.13: Запуск программы lab8-3.asm

2.2 Самостоятельное задание

Написал программу, которая вычисляет сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, где значения x передаются как аргументы. Функция $f(x)$ выбрана из таблицы 8.1 в соответствии с вариантом 8:

$$f(x) = 7 + 2x$$

Программа корректно работает, выводя сумму значений $f(x_1) + f(x_2) + \dots + f(x_n)$. Создал исполняемый файл и проверил его работу на нескольких наборах x (рис. 2.14, рис. 2.15).



```
manieyvazi@VirtualBox: ~/work/
GNU nano 4.8 lab8-prog.asm

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
mov ebx,2
mul ebx
add eax,7
add esi,eax

loop next

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

Рис. 2.14: Программа lab8-prog.asm

```
manleyvazi@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-prog.asm
manleyvazi@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-prog.o -o lab8-p
rog
manleyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-prog 1
f(x)= 7 + 2x
Результат: 9
manleyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-prog 6
f(x)= 7 + 2x
Результат: 19
manleyvazi@VirtualBox:~/work/arch-pc/lab08$ ./lab8-prog 6 9 7 3 4
f(x)= 7 + 2x
Результат: 93
manleyvazi@VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.15: Запуск программы lab8-prog.asm

Программа правильно считает, например, $f(1) = 1$, $f(2) = 5$.

3 Выводы

В ходе работы освоил использование стека, инструкции loop и работу с аргументами командной строки в языке ассемблера NASM.