

Отчёт по лабораторной работе 8

Архитектура компьютеров

Махкамов Рауфджон НММбд-04-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация циклов в NASM	6
2.2	Самостоятельное задание	17
3	Выводы	20

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	15
2.12	Программа lab8-3.asm	16
2.13	Запуск программы lab8-3.asm	17
2.14	Программа lab8-prog.asm	18
2.15	Запуск программы lab8-prog.asm	19

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).

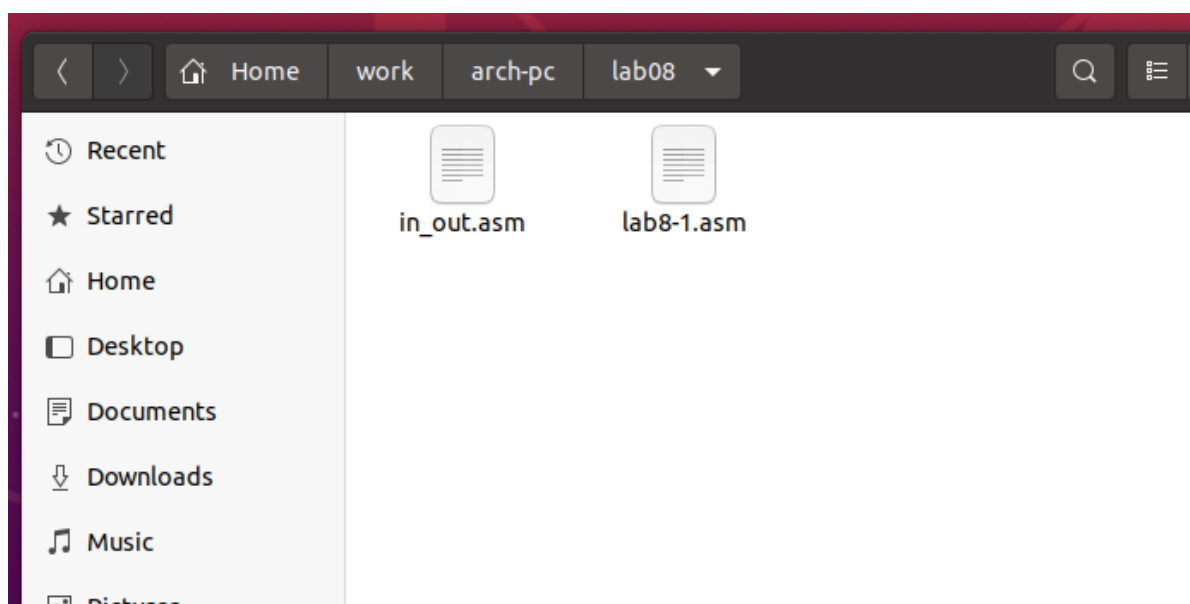


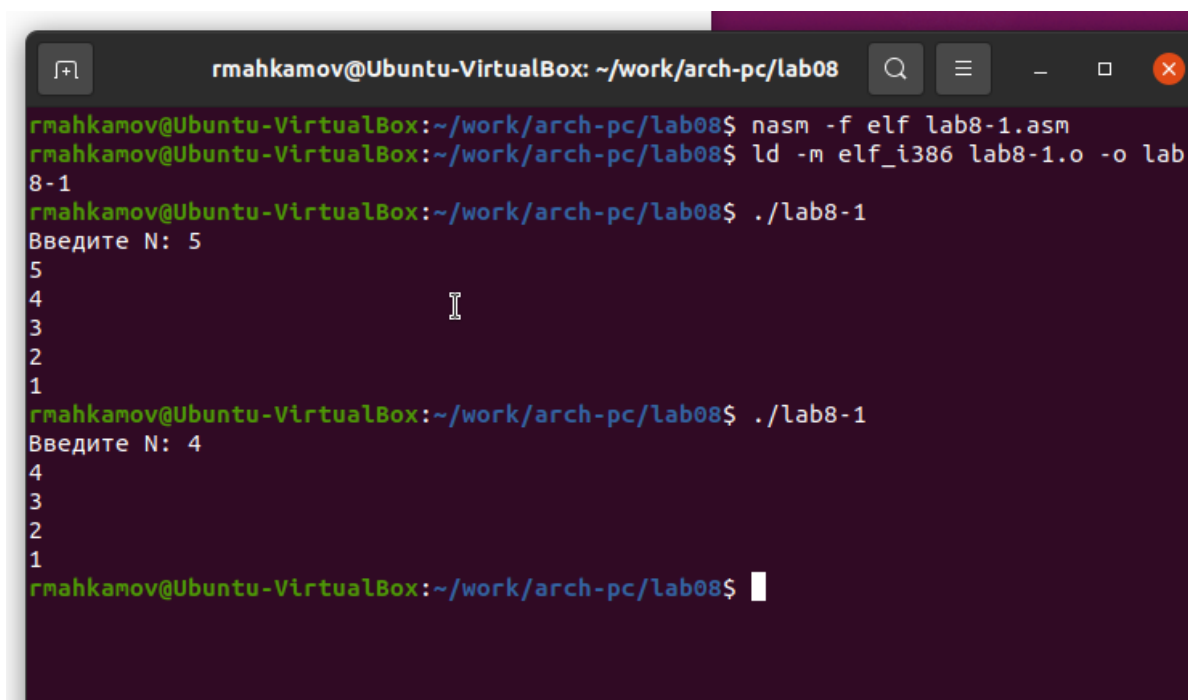
Рис. 2.1: Создан каталог

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).

```
rmahkamov@Ubuntu-VirtualBox: ~/work/arch-pc...
/home/rm~8-1.asm [----] 9 L:[ 1+27 28/ 28] *(636 / 636b) <EO
#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
```

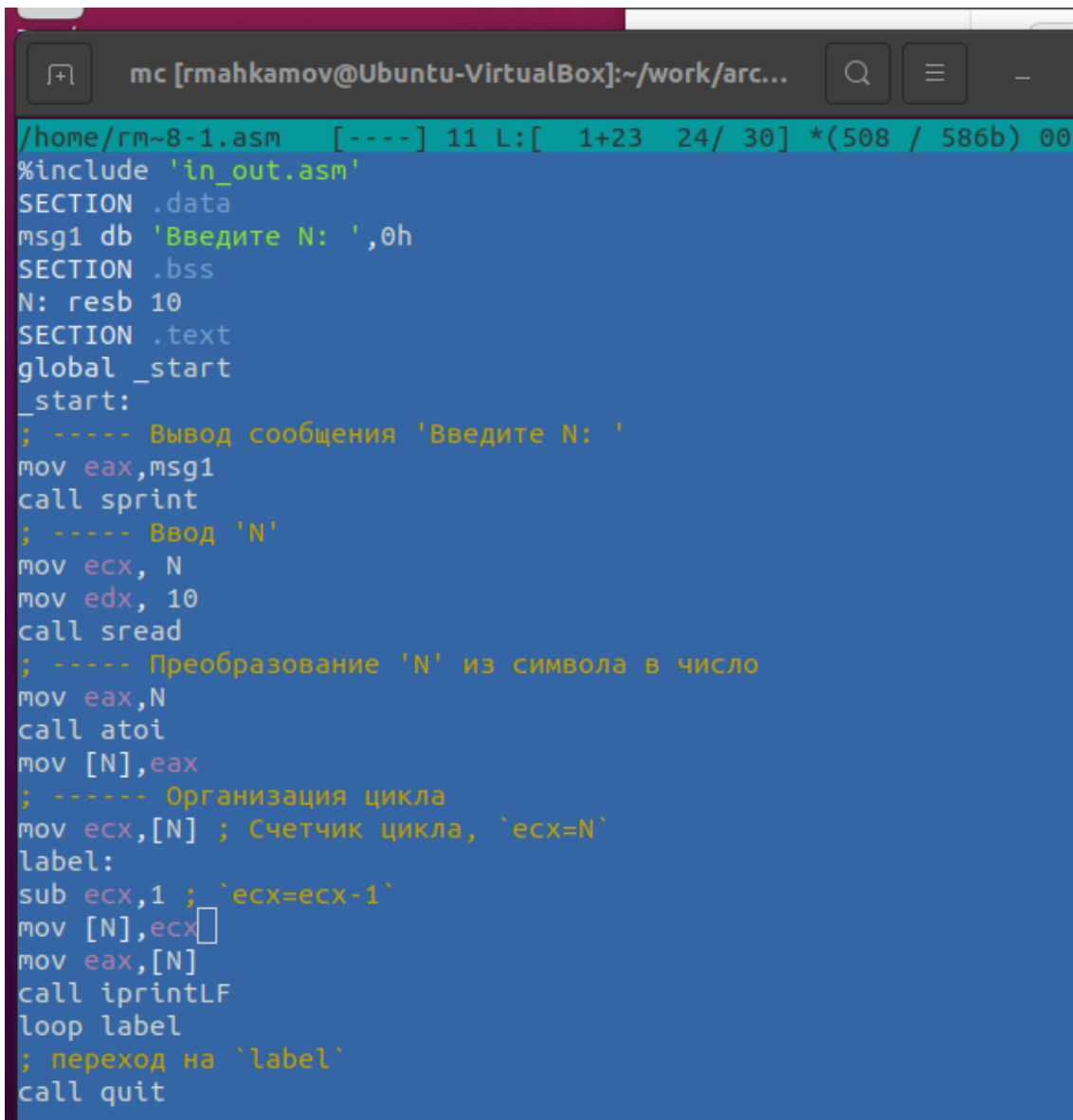
Рис. 2.2: Программа lab8-1.asm



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



```
/home/rm~8-1.asm [----] 11 L: [ 1+23 24/ 30] *(508 / 586b) 00.  
%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

```
4294936958
4294936956
4294936954
4294936952
4294936950
4294936948
4294936946
4294936944
4294936^C
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
rmahkamov@Ubuntu-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 .

```
mc [rmahkamov@Ubuntu-VirtualBox]:~/work/arc...
/home/rm~8-1.asm [----] 11 L:[ 1+10 11/ 31] *(211 / 675b) 001[*]
#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

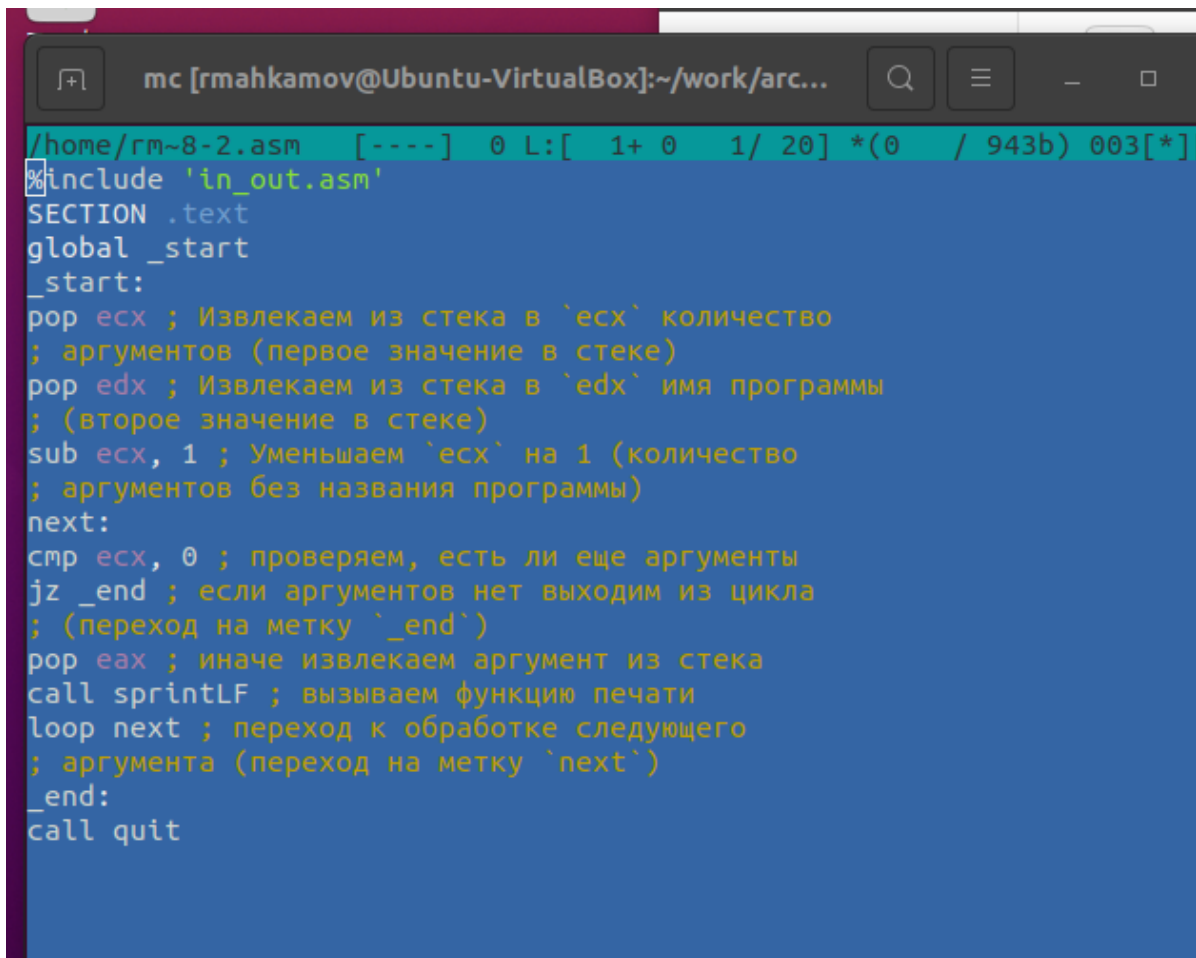
```

rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab
8-1
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$

```

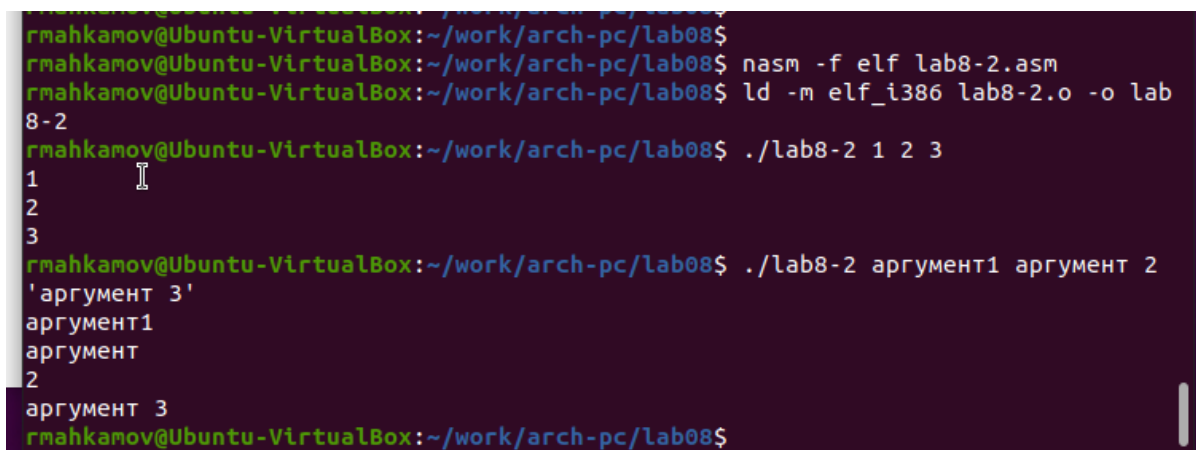
Рис. 2.7: Запуск программы lab8-1.asm

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).



```
mc [rmahkamov@Ubuntu-VirtualBox]:~/work/arc...
/home/rm~8-2.asm [----] 0 L:[ 1+ 0 1/ 20] *(0 / 943b) 003[*]
%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

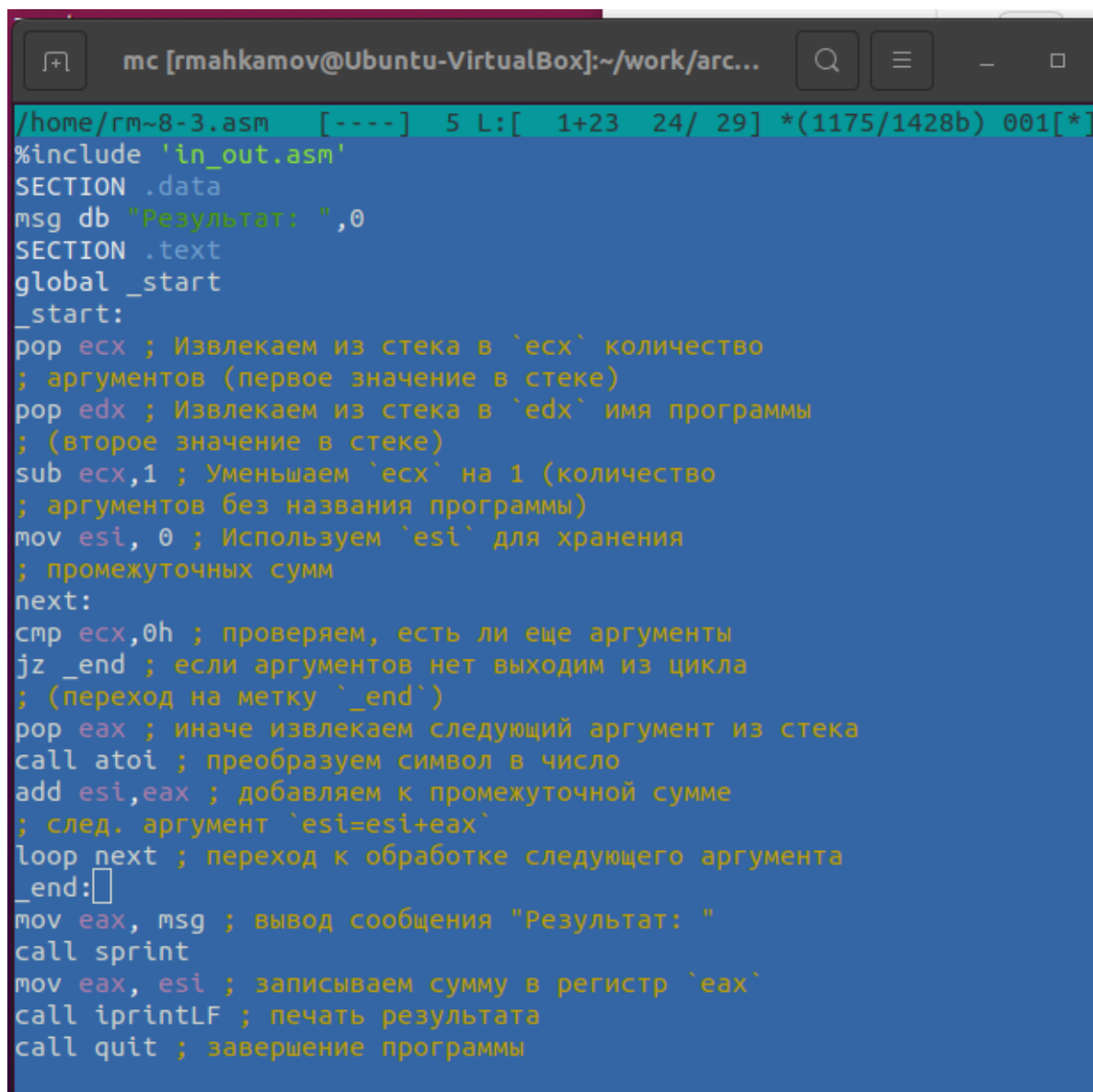


```
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab
8-2
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 1 2 3
1
2
3
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2
'аргумент 3'
аргумент1
аргумент
2
аргумент 3
rmahkamov@Ubuntu-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).



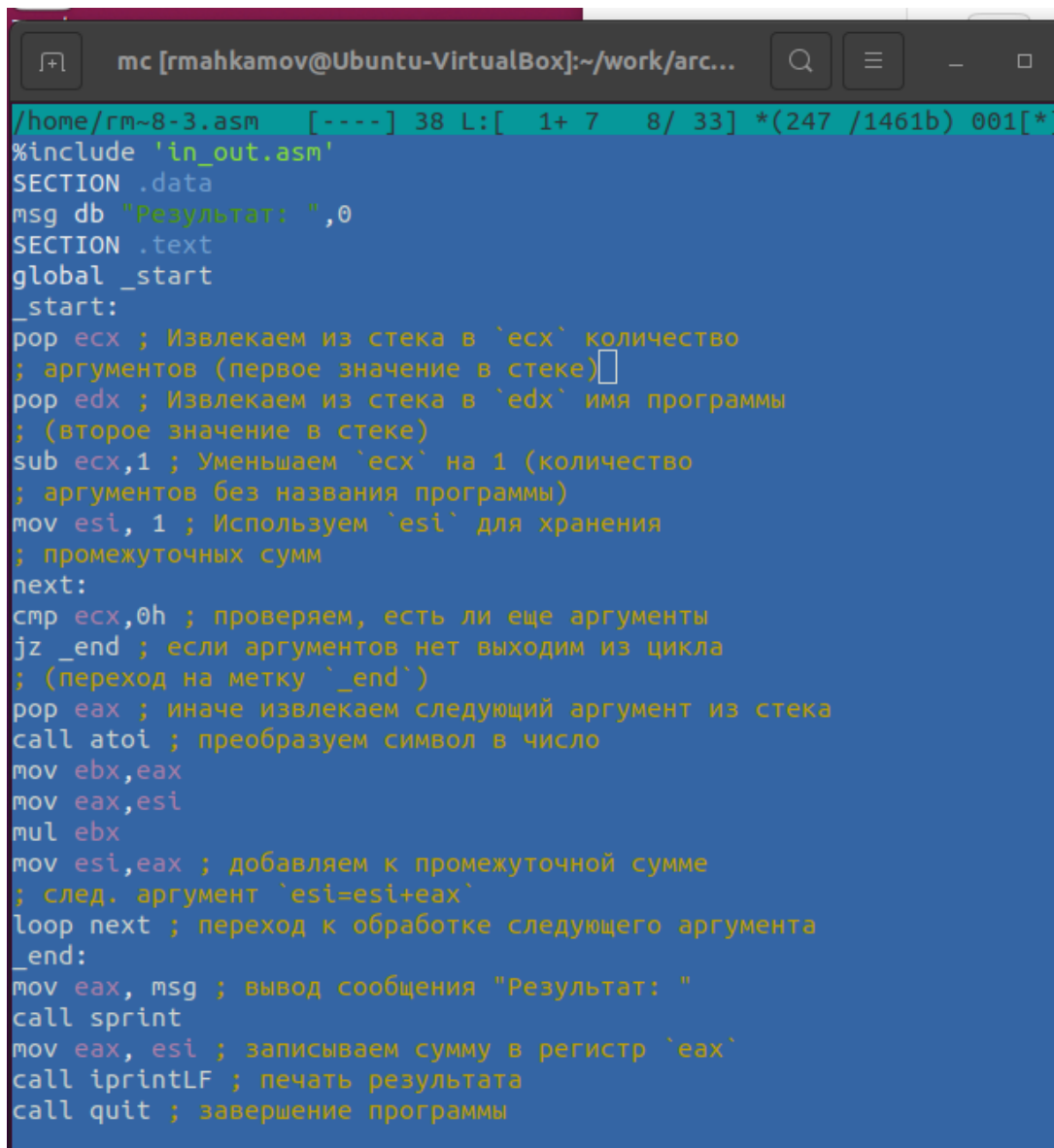
```
/home/rm~8-3.asm [ - - - ] 5 L: [ 1+23 24/ 29 ] *(1175/1428b) 001[*]
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (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. 2.10: Программа lab8-3.asm

```
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab  
8-3  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 10 12 14  
Результат: 36  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 9 8 7  
Результат: 24  
rmahkamov@Ubuntu-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).



```
/home/rm~8-3.asm [----] 38 L:[ 1+ 7 8/ 33] *(247 /1461b) 001[*]
%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
```

Рис. 2.12: Программа lab8-3.asm


```

rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab
8-3
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 10 12 14
Результат: 36
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 9 8 7
Результат: 24
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab
8-3
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 9 8 7
Результат: 504
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 10 12 14
Результат: 1680
rmaHkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$

```

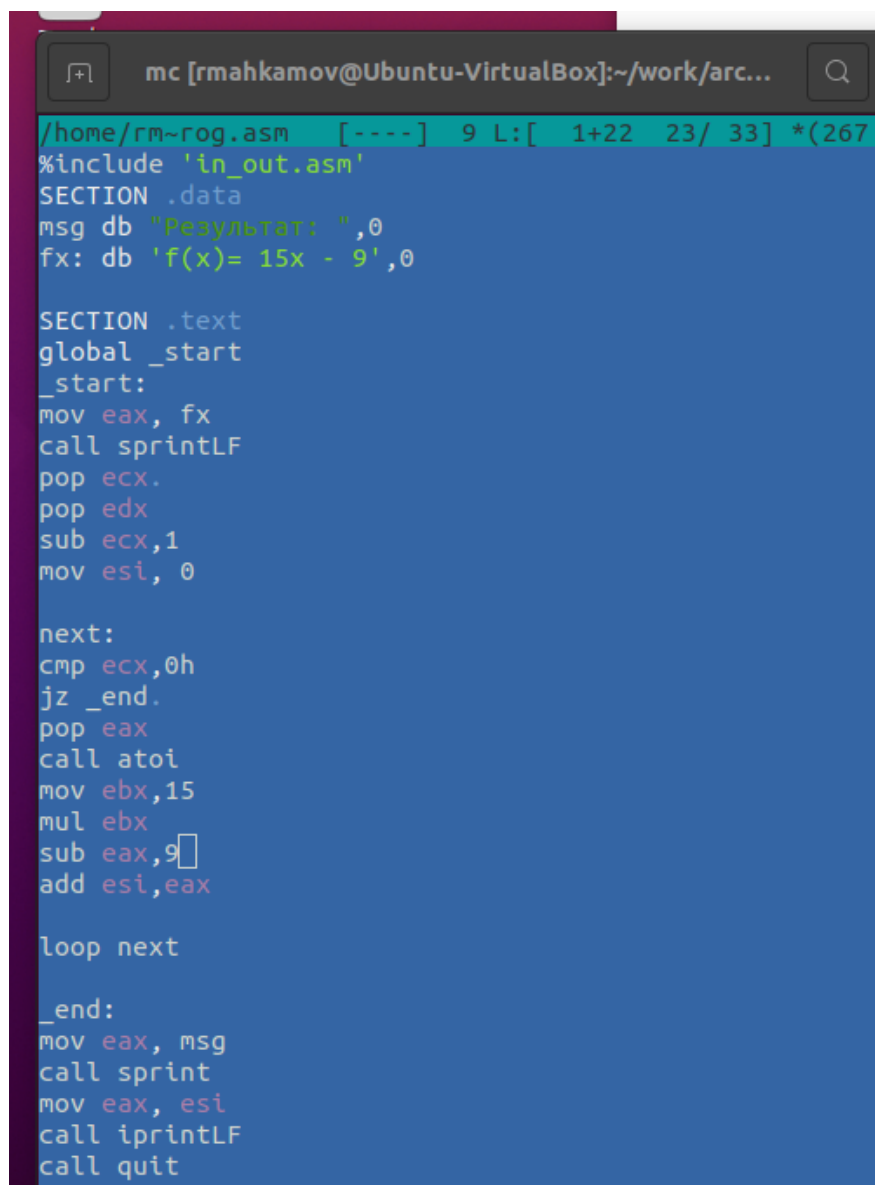
Рис. 2.13: Запуск программы lab8-3.asm

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 12:

$$f(x) = 15x - 9.$$

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).



```
mc [rmahkamov@Ubuntu-VirtualBox]:~/work/arc...  
/home/rm~rog.asm [----] 9 L:[ 1+22 23/ 33] *(267  
%include 'in_out.asm'  
SECTION .data  
msg db "Результат: ",0  
fx: db 'f(x)= 15x - 9',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  
mov ebx,15  
mul ebx  
sub eax,9  
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

```
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-prog.asm  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-prog.o  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-prog.o -o  
lab8-prog  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-prog 1  
f(x)= 15x - 9  
Результат: 6  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-prog 2  
f(x)= 15x - 9  
Результат: 21  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-prog 2 4 5 2 1 3  
f(x)= 15x - 9  
Результат: 201  
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.15: Запуск программы lab8-prog.asm

Программа правильно считает, например, $f(1) = 6$, $f(2) = 21$.

3 Выводы

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