

Шаблон отчёта по лабораторной работе

9

Сильвен Макс Грегор Филс , НКАбд-03-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы :	6
2.1	Реализация циклов в NASM :	6
2.2	Обработка аргументов командной строки :	9
2.3	Программа вычисления суммы аргументов командной строки : .	11
2.4	Выводы по результатам выполнения заданий :	14
3	Задание для самостоятельной работы :	15
3.1	Выводы по результатам выполнения заданий :	17
4	Выводы, согласованные с целью работы :	18
	Список литературы	19

Список иллюстраций

2.1	Ресунок	6
2.2	Ресунок	7
2.3	Ресунок	8
2.4	Ресунок	8
2.5	Ресунок	9
2.6	Ресунок	10
2.7	Ресунок	11
2.8	Ресунок	12
2.9	Ресунок	12
2.10	Ресунок	13
2.11	Ресунок	14
3.1	Ресунок	16
3.2	Ресунок	17

Список таблиц

1 Цель работы

- В девятой лабораторной работе мы научимся писать программы с циклами и обработкой аргументов с помощью командной строки.

2 Выполнение лабораторной работы :

2.1 Реализация циклов в NASM :

- Здесь мы начали с создания каталога для программы лабораторной работы No 9, а затем переместились в девятый каталог лаборатории “~/work/arch-рс/lab09”, после чего мы создали файл “lab9-1.asm”. (рис. 2.1)

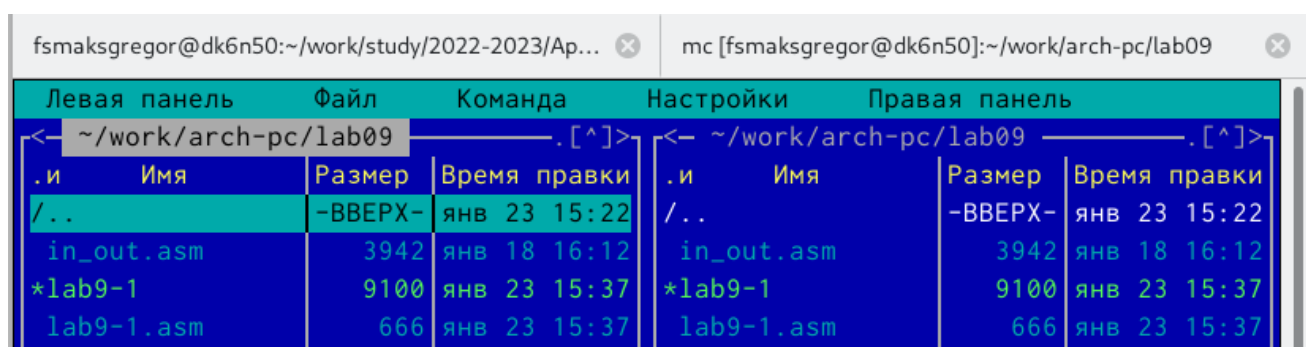
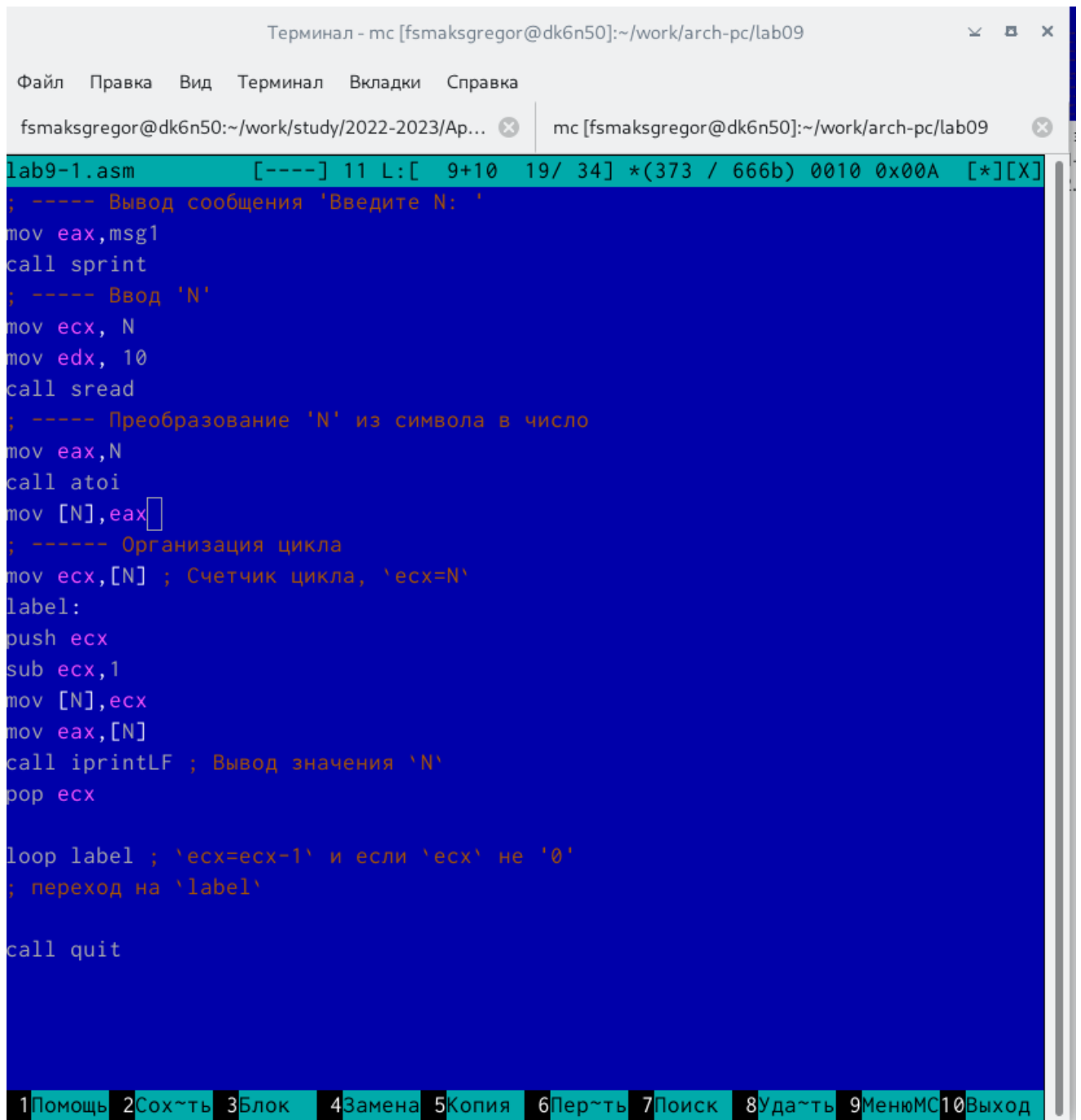


Рис. 2.1: Ресунок

- Затем мы заполнили код нашей программы в файле lab9-1.asm. (рис. 2.2)



Терминал - mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09

Файл Правка Вид Терминал Вкладки Справка

fsmaksgregor@dk6n50:~/work/study/2022-2023/Ар... mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09

```
lab9-1.asm [----] 11 L:[ 9+10 19/ 34] *(373 / 666b) 0010 0x00A [*][X]
; ----- 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
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения 'N'
pop ecx

loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'

call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню MS 10Выход

Рис. 2.2: Ресунок

- После этого мы скомпилировали файл, создали исполняемый файл и проверили его работу.(рис. 2.3)

```
fsmaksgregor@dk6n50:~/work/... x mc [fsmaksgregor@dk6n50]:~/... x fsmaksgregor@dk6n50:~/work/... x
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $
```

Рис. 2.3: Рисунок

- Мы внесли изменения в наш код, а затем создали исполняемый файл.(рис. 2.4)

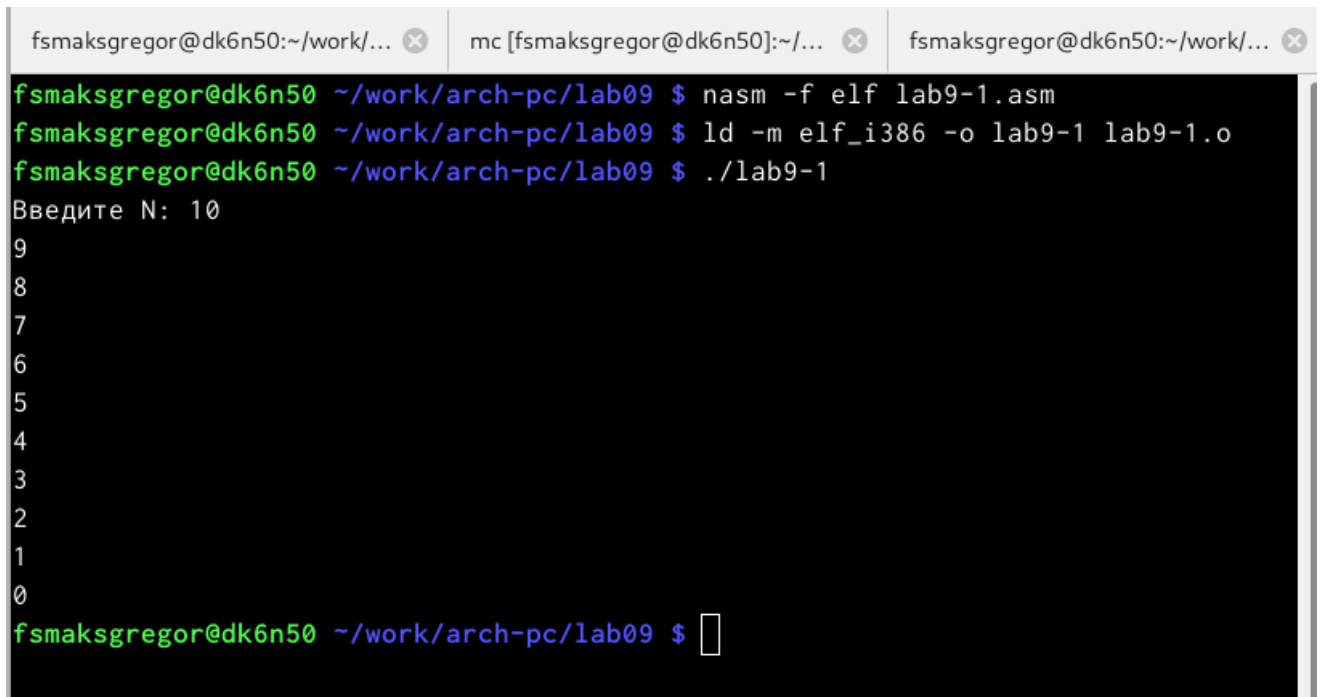
```
fsmaksgregor@dk6n50:~/work/... x mc [fsmaksgregor@dk6n50]:~/... x fsmaksgregor@dk6n50:~/work/... x
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 10
9
7
5
3
1
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $
```

Рис. 2.4: Рисунок

- Регистр ехх принимает пять значений, которые являются: 9,7,5,3,1, мы можем заметить, что количество циклов не соответствует числу, введенному

пользователем

- На этот раз мы использовали стек, и в конечном итоге количество циклов соответствует числу, которое было введено в начале.(рис. 2.5)

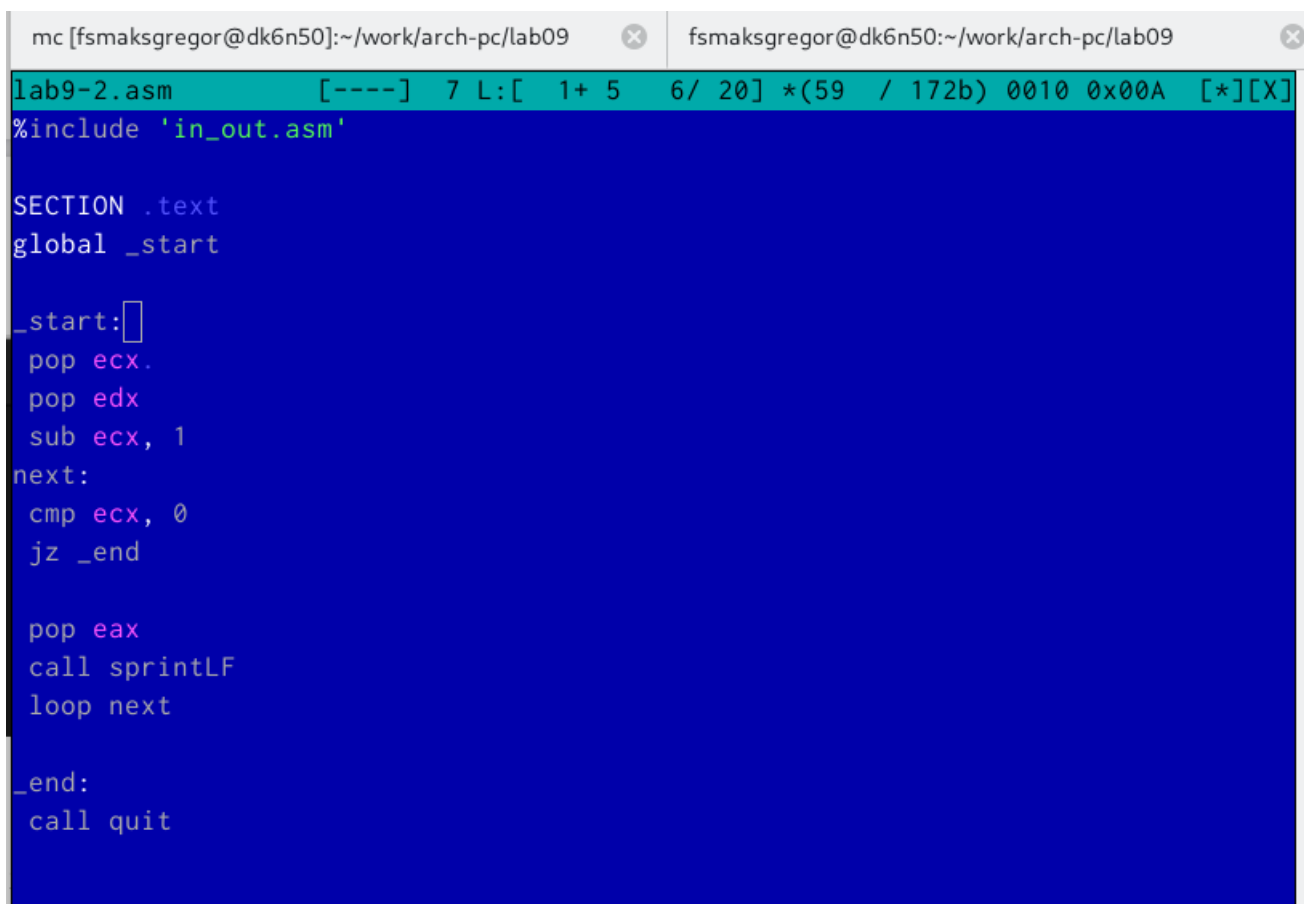


```
fsmaksgregor@dk6n50:~/work/... x mc [fsmaksgregor@dk6n50]:~/... x fsmaksgregor@dk6n50:~/work/... x
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $
```

Рис. 2.5: Ресунок

2.2 Обработка аргументов командной строки :

- На этом шаге мы создали файл lab9-2.asm, затем заполнили в нем наш код.(рис. 2.6)



```
lab9-2.asm [----] 7 L: [ 1+ 5 6/ 20] *(59 / 172b) 0010 0x00A [*][X]
#include 'in_out.asm'

SECTION .text
global _start

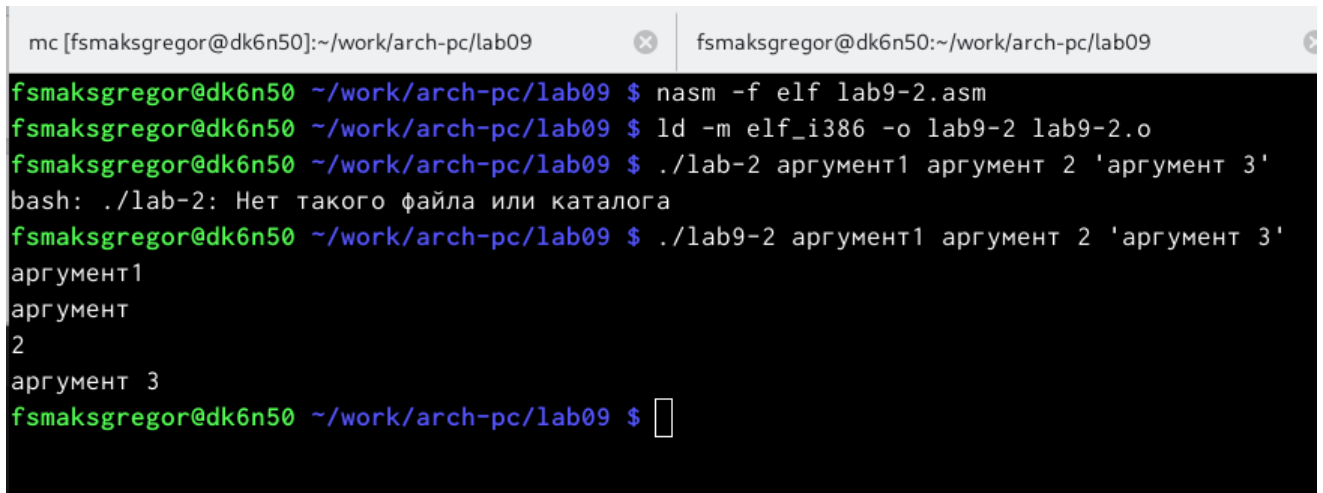
_start:
    pop ecx
    pop edx
    sub ecx, 1
next:
    cmp ecx, 0
    jz _end

    pop eax
    call sprintLF
    loop next

_end:
    call quit
```

Рис. 2.6: Рисунок

- После этого мы скомпилировали файл и создали исполняемый файл.(рис. 2.7)



```
mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09  fsmaksgregor@dk6n50:~/work/arch-pc/lab09
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ nasm -f elf lab9-2.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./lab-2 аргумент1 аргумент 2 'аргумент 3'
bash: ./lab-2: Нет такого файла или каталога
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $
```

Рис. 2.7: Ресунок

- И, как вы можете видеть, на этот раз при запуске программы мы добавили в команду три аргумента, и в этом случае были обработаны три аргумента.

2.3 Программа вычисления суммы аргументов командной строки :

- Первым делом мы создали файл lab9-3.asm, затем заполнили кодом программы.(рис. 2.8)

```

lab9-3.asm      [----] 36 L:[ 1+11 12/ 30] *(508 /1429b) 0010 0x00A  [*][X]
#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

```

Рис. 2.8: Ресунок

- После этого мы скомпилировали файл, затем создали исполняемый файл, ввели нужное количество аргументов и запустили program.(рис. 2.9)

```

mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09  fsmaksgregor@dk6n50:~/work/arch-pc/lab09
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ nasm -f elf lab9-3.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./lab9-3 12 13 7 10 5
Результат: 47
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ 

```

Рис. 2.9: Ресунок

- Затем мы изменили код, чтобы вычислить произведение аргументов командной строки.(рис. 2.10)

```

Терминал - mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09
Файл  Правка  Вид  Терминал  Вкладки  Справка
fsmaksgregor@dk6n50:~/work/study/2022-2023/Ар...  mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09
lab9-3.asm  [----]  5 L:[  4+13  17/ 34] *(695 /1463b) 0110 0x06E  [*][X]
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
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюMC10Выход

```

Рис. 2.10: Рисунок

- После этого я скомпилировал код и запустил исполняемый файл.(рис. 2.11)

```
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ nasm -f elf lab9-3.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./lab9-3 1 2 3 4 5
Результат: 120
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $
```

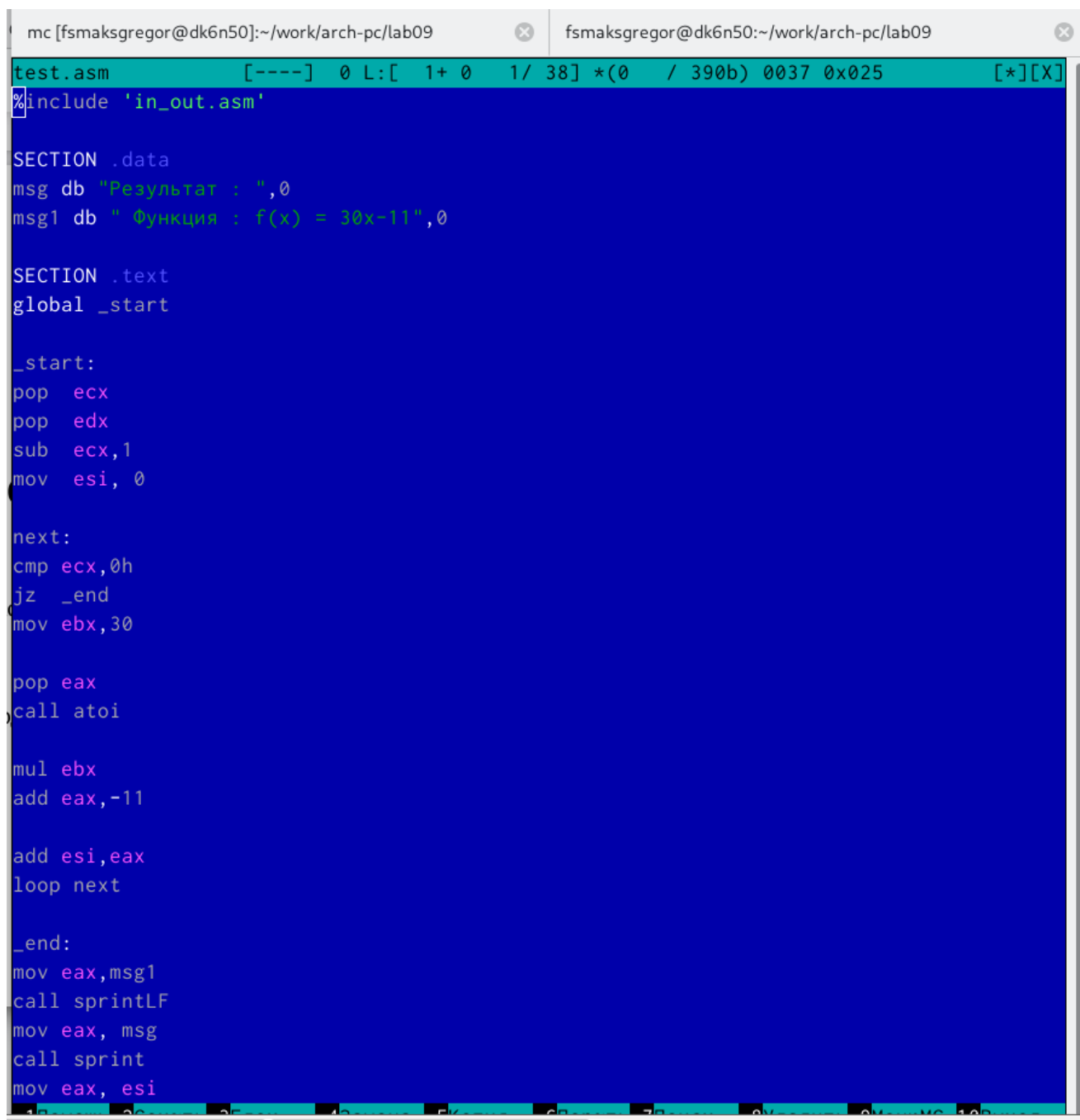
Рис. 2.11: Ресунок

2.4 Выводы по результатам выполнения заданий :

- В этой части работы мы узнали, как манипулировать циклами, как правильно использовать стек для написания программ

3 Задание для самостоятельной работы :

- В этой части мы должны были написать программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x$
- сначала мы создали наш файл `test.asm`, где будет находиться наш код, затем мы написали программу. (рис. 3.1)



```
mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09  fsmaksgregor@dk6n50:~/work/arch-pc/lab09
test.asm [----] 0 L:[ 1+ 0 1/ 38] *(0 / 390b) 0037 0x025 [*][X]
%include 'in_out.asm'

SECTION .data
msg db "Результат : ",0
msg1 db " Функция : f(x) = 30x-11",0

SECTION .text
global _start

_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
mov ebx,30

pop eax
call atoi

mul ebx
add eax,-11

add esi,eax
loop next

_end:
mov eax,msg1
call sprintLF
mov eax, msg
call sprint
mov eax, esi
```

Рис. 3.1: Ресунок

- Затем мы протестировали нашу программу.(рис. 2.1)



```
mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab09  fsmaksgregor@dk6n50:~/work/arch-pc/lab09
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ nasm -f elf test.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o test test.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $ ./test 1 2 3 4
Функция :  $f(x) = 30x - 11$ 
Результат : 256
fsmaksgregor@dk6n50 ~/work/arch-pc/lab09 $
```

Рис. 3.2: Ресунок

3.1 Выводы по результатам выполнения заданий :

В этой части мы узнали, как вычислить сложную математическую операцию, которая имеет функции, используя циклы и стек.

4 Выводы, согласованные с целью работы :

- В девятой лабораторной работе мы узнали, как использовать циклы и стек в NASM.

Список литературы