

Отчёт по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Кичигина Полина Евгеньевна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация циклов в NASM	6
3.2	Обработка аргументов командной строки.	10
3.3	Задание для самостоятельной работы	13
4	Выводы	16

Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
3.2	Заполняем файл	7
3.3	Запускаем файл и проверяем его работу	7
3.4	Изменяем файл	8
3.5	Запускаем файл и смотрим на его работу	8
3.6	Редактируем файл	9
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	9
3.8	Создаем файл командой <code>touch</code>	10
3.9	Заполняем файл	11
3.10	Смотрим на работу программ	11
3.11	Создаем файл командой <code>touch</code>	12
3.12	Заполняем файл	12
3.13	Смотрим на работу программы	12
3.14	Изменяем файл	13
3.15	Проверяем работу файла(работает правильно)	13
3.16	Создаем файл командой <code>touch</code>	14
3.17	Пишем программу	14
3.18	Смотрим на работу программы при $x_1=5$ $x_2=3$ $x_3=4$ $x_4=5$ (всё верно)	15

1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

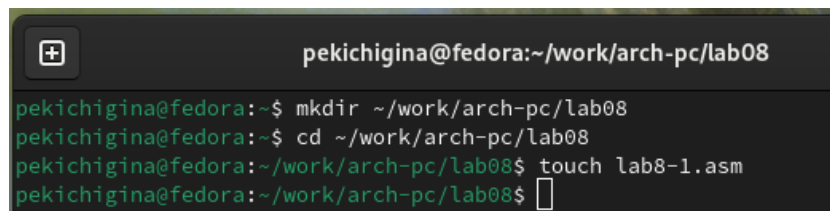
2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

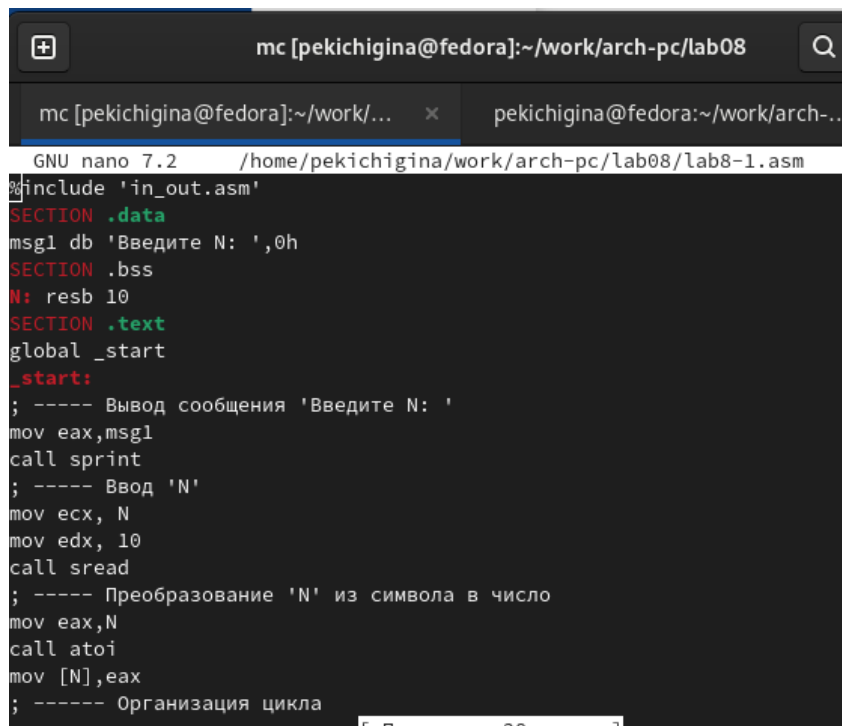
Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. fig. 3.1)

A terminal window with a dark background. The title bar shows a plus icon and the text 'pekichigina@fedora:~/work/arch-pc/lab08'. The terminal contains the following commands and their outputs:

```
pekichigina@fedora:~$ mkdir ~/work/arch-pc/lab08
pekichigina@fedora:~$ cd ~/work/arch-pc/lab08
pekichigina@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
pekichigina@fedora:~/work/arch-pc/lab08$
```

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

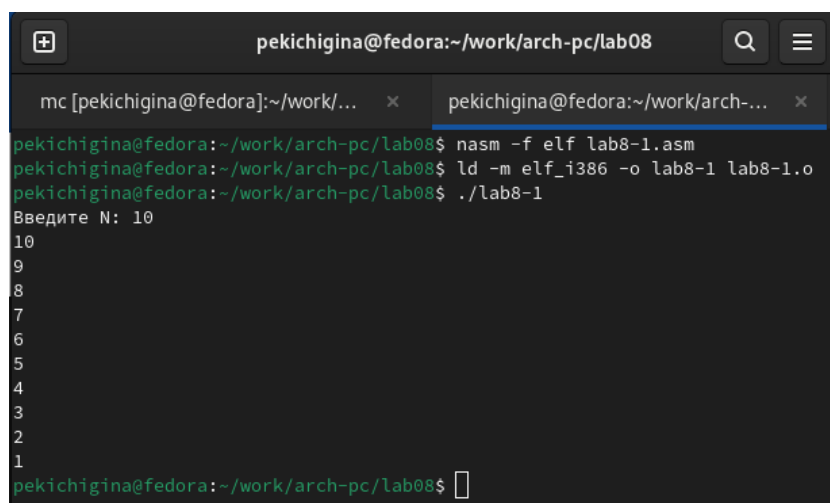
Открываем файл в nano и заполняем его в соответствии с листингом 8.1 (рис. fig. 3.2)



```
mc [pekichigina@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/pekichigina/work/arch-pc/lab08/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
```

Рис. 3.2: Заполняем файл

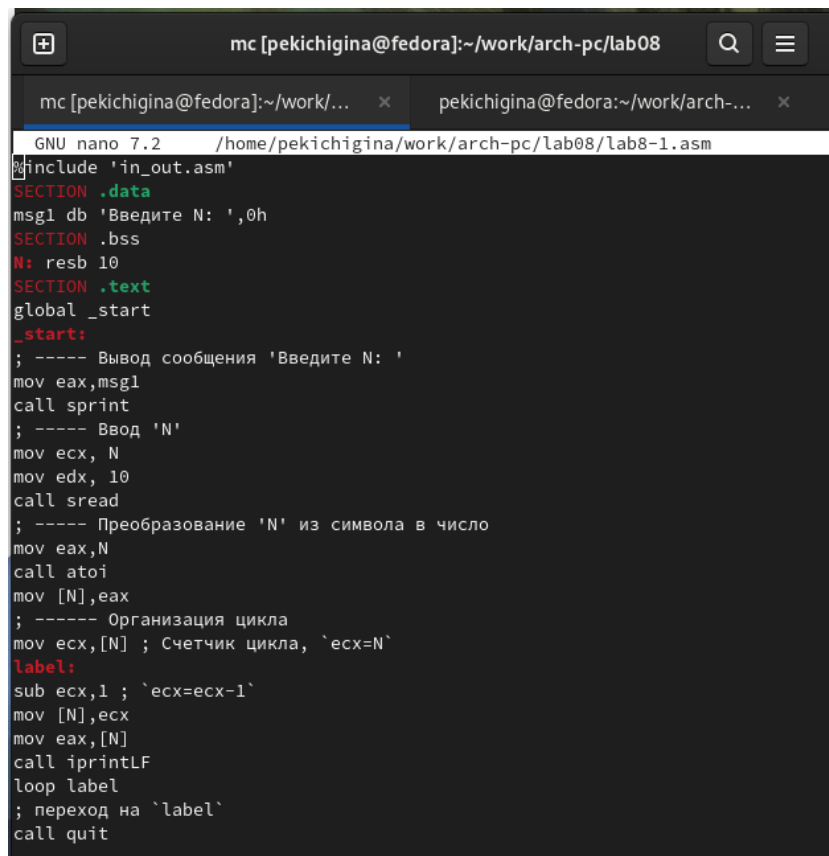
Создаем исполняемый файл и запускаем его (рис. fig. 3.3)



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

Рис. 3.3: Запускаем файл и проверяем его работу

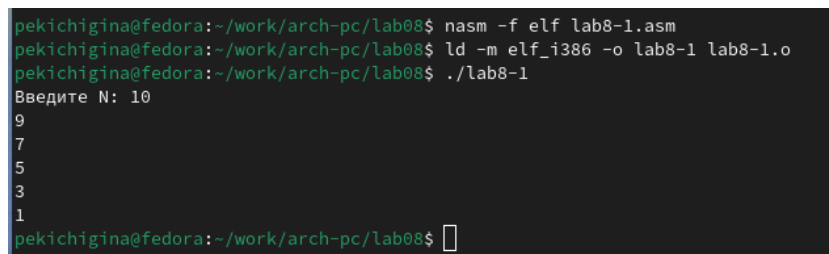
Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. fig. 3.4)



```
mc [pekichigina@fedora]:~/work/arch-pc/lab08
mc [pekichigina@fedora]:~/work/... x pekichigina@fedora:~/work/arch-... x
GNU nano 7.2 /home/pekichigina/work/arch-pc/lab08/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. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5)



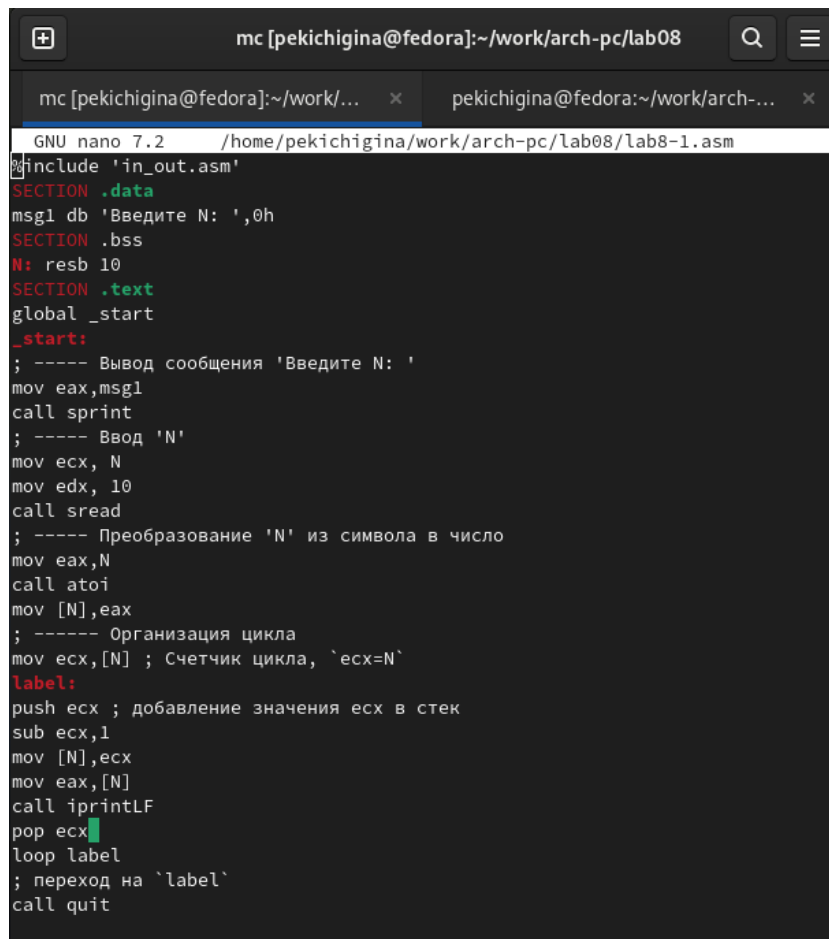
```
pekichigina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
pekichigina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
pekichigina@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
pekichigina@fedora:~/work/arch-pc/lab08$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Регистр `ecx` принимает значения 9,7,5,3,1(на вход подается число 10, в цикле `label` данный регистр уменьшается на 2 командой `sub` и `loop`).

Число проходов цикла не соответствует числу `N`, так как уменьшается на 2.

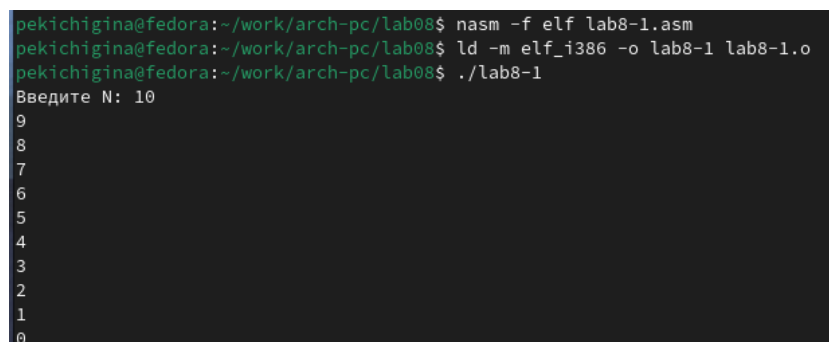
Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис. fig. 3.6)



```
mc [pekichigina@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/pekichigina/work/arch-pc/lab08/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
loop label
; переход на `label`
call quit
```

Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.7)



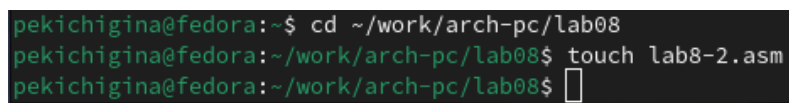
```
pekichigina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
pekichigina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
pekichigina@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

В данном случае число проходов цикла равна числу N.

3.2 Обработка аргументов командной строки.

Создаем новый файл (рис. fig. 3.8)



```
pekichigina@fedora:~$ cd ~/work/arch-pc/lab08
pekichigina@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
pekichigina@fedora:~/work/arch-pc/lab08$
```

Рис. 3.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. fig. 3.9)

```

GNU nano 7.2 /home/pekichigina/work/arch-pc/lab08/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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
    _end:
    call quit

```

Имя файла для записи: /home/pekichigina/work/arch-pc/lab08/lab8-2.o

^G Справка	M-D Формат DOS	M-A Доп. в начало	M-~
^C Отмена	M-M Формат Mac	M-P Доп. в конец	^_

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. fig. 3.10)

```

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

```

Рис. 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm (рис. fig. 3.11)

```
pekichigina@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
pekichigina@fedora:~/work/arch-pc/lab08$
```

Рис. 3.11: Создаем файл командой touch

Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. fig. 3.12)

```
GNU nano 7.2 /home/pekichigina/work/arch-pc/lab08/lab8-3.asm Изменён
#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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выворнять ^/_ К строке

Рис. 3.12: Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.13)

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

Рис. 3.13: Смотрим на работу программы

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. fig. 3.14)

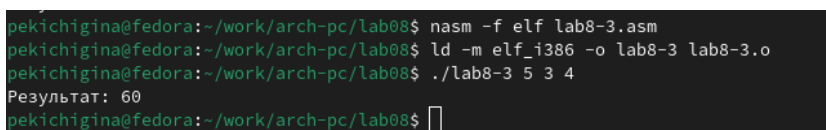


```
GNU nano 7.2 /home/pekichigina/work/arch-pc/lab08/lab8-3.asm Изменён
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (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
mul esi
mov esi,eax
loop next
_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

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выворнять ^/_ К строке
```

Рис. 3.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.15)



```
pekichigina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
pekichigina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
pekichigina@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 3 4
Результат: 60
pekichigina@fedora:~/work/arch-pc/lab08$
```

Рис. 3.15: Проверяем работу файла(работает правильно)

3.3 Задание для самостоятельной работы

1. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1)+f(x_2)+\dots+f(x_n)$.

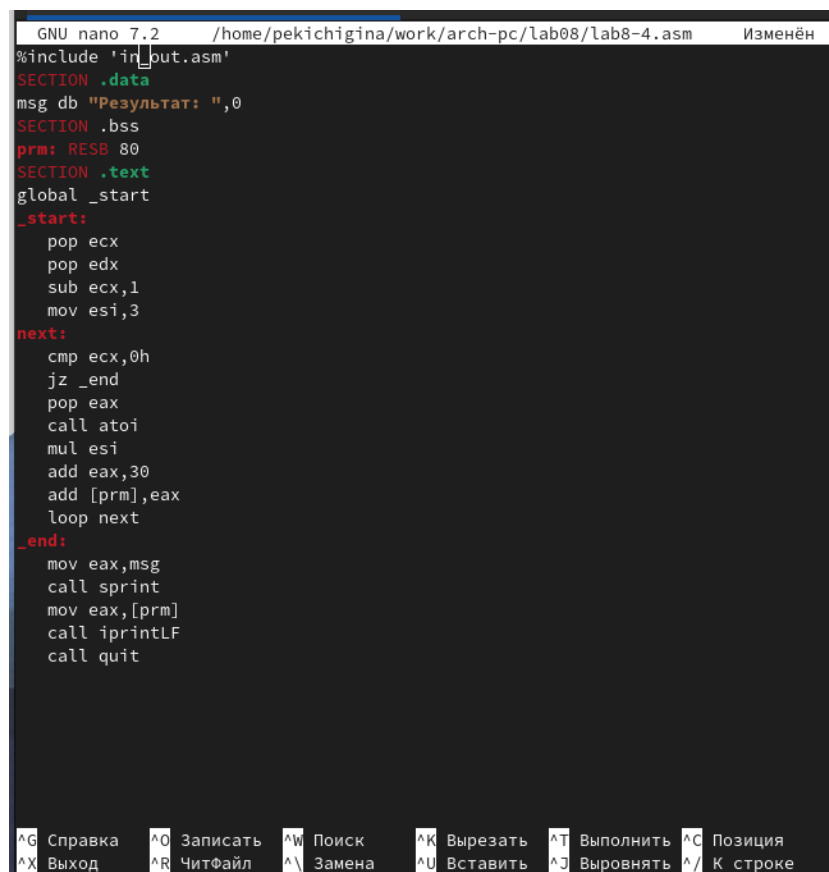
Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

Создаем новый файл (рис. fig. 3.16)

```
pekichigina@fedora:~/work/arch-pc/lab08$ touch lab8-4.asm
pekichigina@fedora:~/work/arch-pc/lab08$
```

Рис. 3.16: Создаем файл командой touch

Открываем его и пишем программу, которая выведет сумму значений, полученных после решения выражения $3(10+x)$ (рис. fig. 3.17)



```
GNU nano 7.2 /home/pekichigina/work/arch-pc/lab08/lab8-4.asm Изменён
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .bss
prm: RESB 80
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi,3
next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    mul esi
    add eax,30
    add [prm],eax
    loop next
_end:
    mov eax,msg
    call sprint
    mov eax,[prm]
    call iprintLF
    call quit
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/_ К строке

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. fig. 3.18)

```
pekichigina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
pekichigina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
pekichigina@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4 5
Результат: 195
pekichigina@fedora:~/work/arch-pc/lab08$
```

Рис. 3.18: Смотрим на рабботу программы при $x_1=5$ $x_2=3$ $x_3=4$ $x_4=5$ (всё верно)

4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.