

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Баранов Никита Дмитриевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Самостоятельная работа	14
5	Выводы	16

Список иллюстраций

3.1	Создаем каталоги и файл	7
3.2	Вводим программу в файл	8
3.3	Создаем объектный файл и проверяем работу программы	8
3.4	Редактируем файл. Изменяем программу	9
3.5	Создаем объектный файл и проверяем работу измененной программы	9
3.6	Изменяем программу, добавляя стек	10
3.7	Создаем объектный файл и проверяем его работу	10
3.8	Создаем файл и вводим программу	11
3.9	Создаем объектный файл и проверяем работу программы с аргументами	11
3.10	Создаем файл и вводим программу	12
3.11	Создаем объектный файл и проверяем работу программы с аргументами	12
3.12	Изменяем программу, чтобы она выводила произведение	13
3.13	Создаем объектный файл и проверяем работу программы	13
4.1	Создаем файл и вписываем программу N18	15
4.2	Создаем объектный файл и проверяем работу программы на нескольких примерах	15

Список таблиц

1 Цель работы

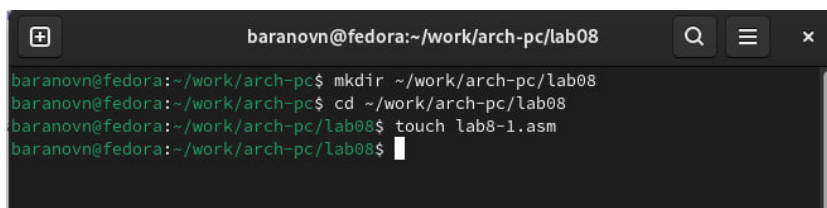
Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки

2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

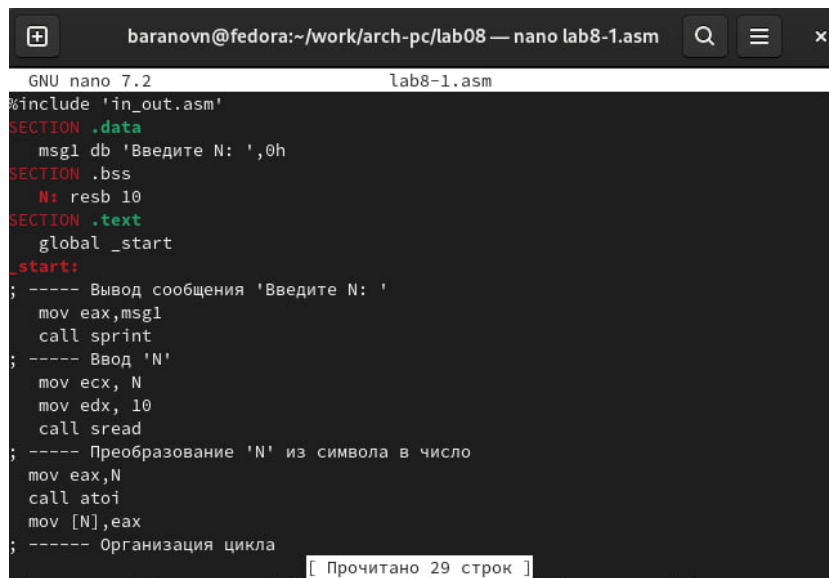
3 Выполнение лабораторной работы

Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.Измените текст программы добавив изменение значение регистра esx в цикле. Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр esx в цикле? Соответствует ли число проходов цикла значению N введенному с клавиатуры?(рис. fig. 3.1)(рис. fig. 3.2)(рис. fig. 3.3)(рис. fig. 3.4)(рис. fig. 3.5).

A terminal window with a dark background. The title bar shows the user 'baranovn@fedora' and the current directory '~/work/arch-pc/lab08'. The terminal contains the following commands and their outputs:

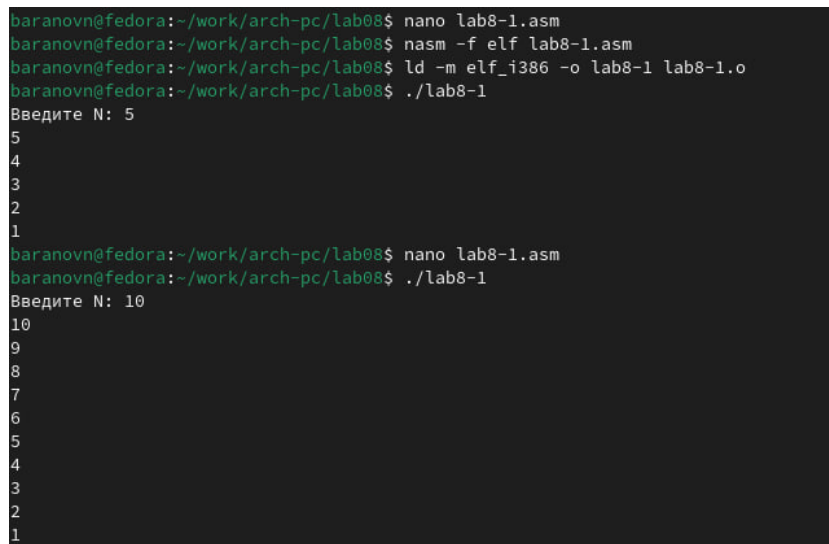
```
baranovn@fedora:~/work/arch-pc$ mkdir ~/work/arch-pc/lab08
baranovn@fedora:~/work/arch-pc$ cd ~/work/arch-pc/lab08
baranovn@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
baranovn@fedora:~/work/arch-pc/lab08$
```

Рис. 3.1: Создаем каталоги и файл



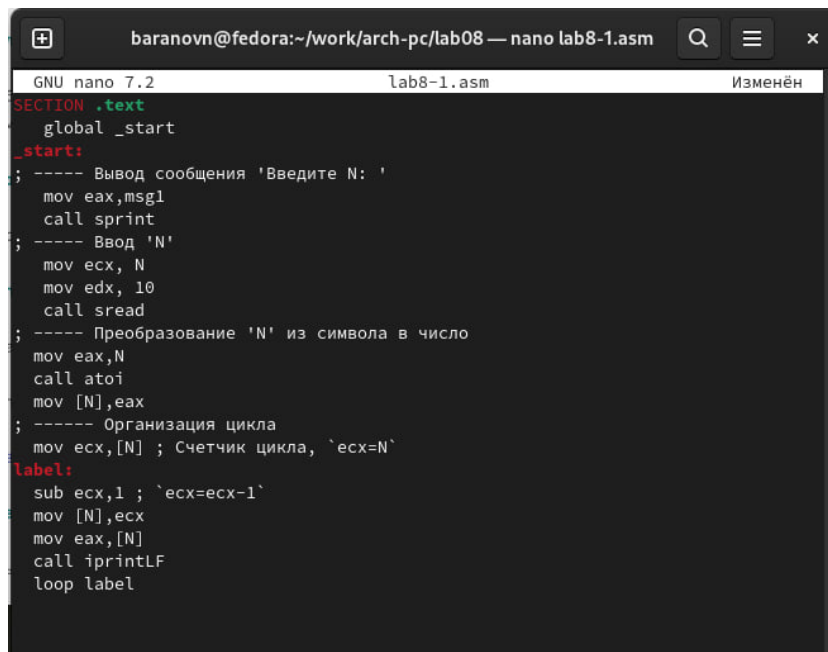
```
GNU nano 7.2 lab8-1.asm
%include 'in_out.asm'
SECTION .data
    msg1 db 'Введите N: ',0h
SECTION .bss
    N: resb 10
SECTION .text
    global _start
_start:
; ----- Вывод сообщения 'Введите N: '
    mov eax,msg1
    call sprint
; ----- Ввод 'N'
    mov ecx, N
    mov edx, 10
    call sread
; ----- Преобразование 'N' из символа в число
    mov eax,N
    call atoi
    mov [N],eax
; ----- Организация цикла
[ Прочитано 29 строк ]
```

Рис. 3.2: Вводим программу в файл



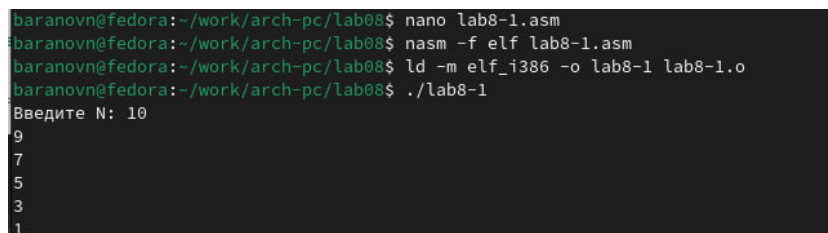
```
baranovn@fedora:~/work/arch-pc/lab08$ nano lab8-1.asm
baranovn@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
baranovn@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
baranovn@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
baranovn@fedora:~/work/arch-pc/lab08$ nano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

Рис. 3.3: Создаем объектный файл и проверяем работу программы



```
baranovn@fedora:~/work/arch-pc/lab08 — nano lab8-1.asm
GNU nano 7.2 lab8-1.asm Изменён
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 iprintfLF
loop label
```

Рис. 3.4: Редактируем файл. Изменяем программу



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

Рис. 3.5: Создаем объектный файл и проверяем работу измененной программы

Регистру `ecx` присваиваются значения 9 7 5 3 1 - регистр уменьшается на 2. Число проходов не соответствует числу `N`, из-за уменьшения на 2.

Внесите изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению `N` введенному с клавиатуры?(рис. fig. 3.6)(рис. fig. 3.7)

```

baranovn@fedora:~/work/arch-pc/lab08 — nano lab8-1.asm
GNU nano 7.2 lab8-1.asm
; ----- 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
^O Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^_ К строке

```

Рис. 3.6: Изменяем программу, добавляя стек

```

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

```

Рис. 3.7: Создаем объектный файл и проверяем его работу

В измененной программе число проходов соответствует числу N.

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2. Создайте исполняемый файл и запустите его, указав аргументы. Сколько аргументов было обработано программой?(рис. fig. 3.8)(рис. fig. 3.9)

```

GNU nano 7.2 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
  
```

Рис. 3.8: Создаем файл и вводим программу

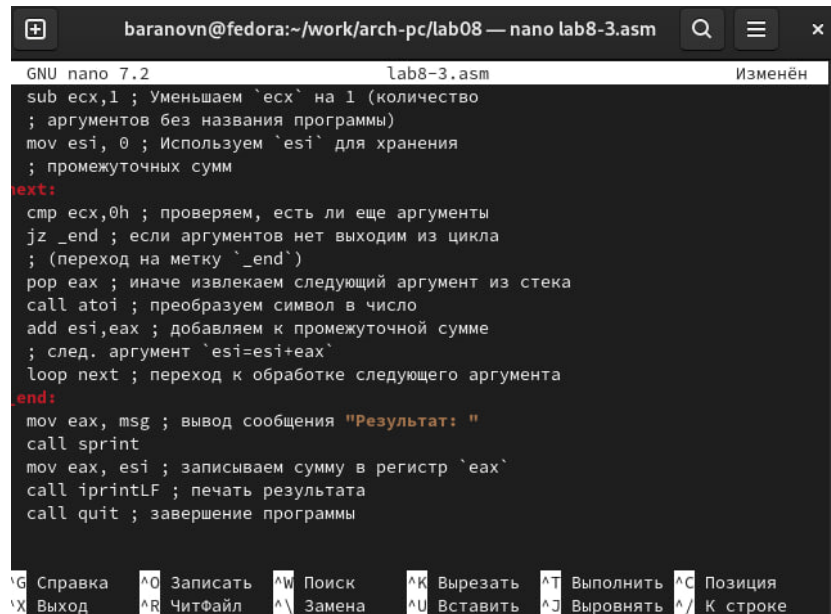
```

baranov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
baranov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
baranov@fedora:~/work/arch-pc/lab08$ ./lab8-2 5 0 3
5
0
3
baranov@fedora:~/work/arch-pc/lab08$
  
```

Рис. 3.9: Создаем объектный файл и проверяем работу программы с аргументами

Программа обрабатывает 3 аргумента

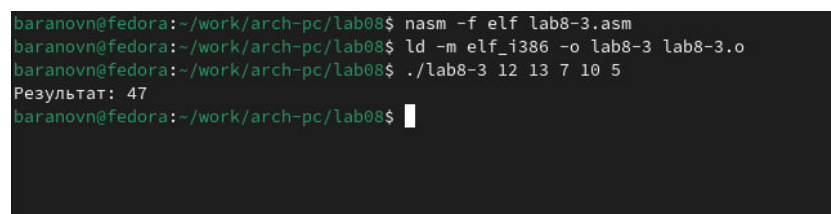
Создайте файл lab8-3.asm в каталоге ~/work/archpc/lab08 и введите в него текст программы из листинга 8.3. Создайте исполняемый файл и запустите его, указав аргументы. Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. fig. 3.10)(рис. fig. 3.11)(рис. fig. 3.12)(рис. fig. 3.13)



```
GNU nano 7.2 lab8-3.asm
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

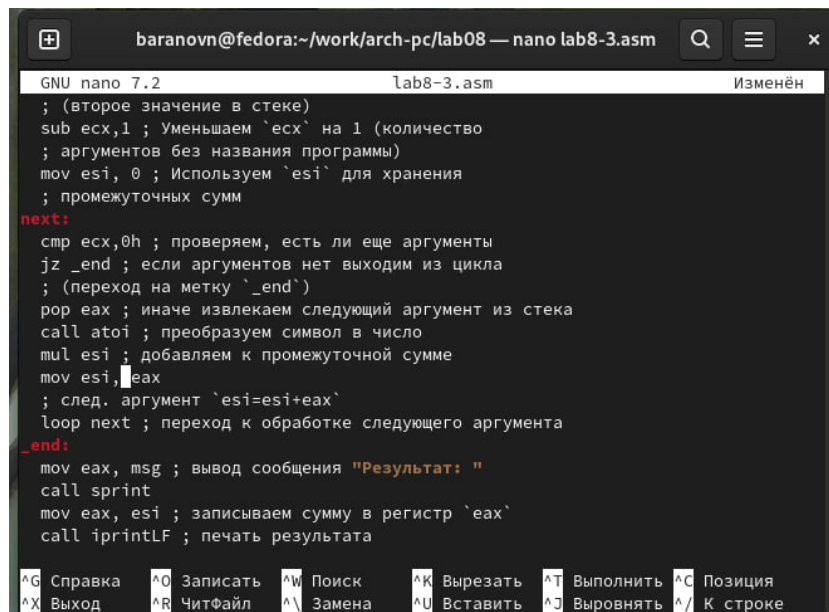
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^_/ К строке
```

Рис. 3.10: Создаем файл и вводим программу



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

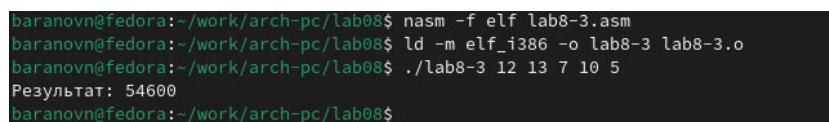
Рис. 3.11: Создаем объектный файл и проверяем работу программы с аргументами



```
GNU nano 7.2 lab8-3.asm Изменён
; (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 ; добавляем к промежуточной сумме
mov esi,eax
; 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

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

Рис. 3.12: Изменяем программу, чтобы она выводила произведение

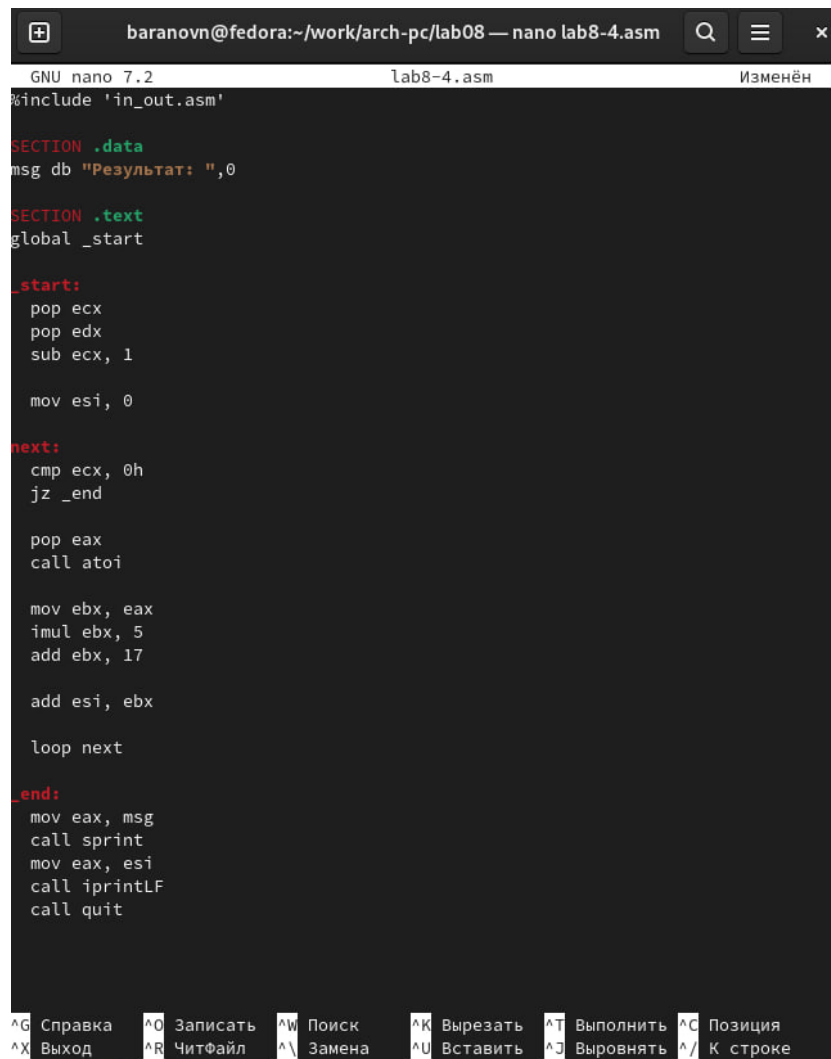


```
baranov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
baranov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
baranov@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
baranov@fedora:~/work/arch-pc/lab08$
```

Рис. 3.13: Создаем объектный файл и проверяем работу программы

4 Самостоятельная работа

Напишите программу, которая находит сумму значений функции $\varphi(x)$ 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 $x=x_1, x_2, x_n$ (рис. fig. 4.1)(рис. fig. 4.2)



```
baranovn@fedora:~/work/arch-pc/lab08 — nano lab8-4.asm
GNU nano 7.2 lab8-4.asm Изменён
#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, eax
    imul ebx, 5
    add ebx, 17

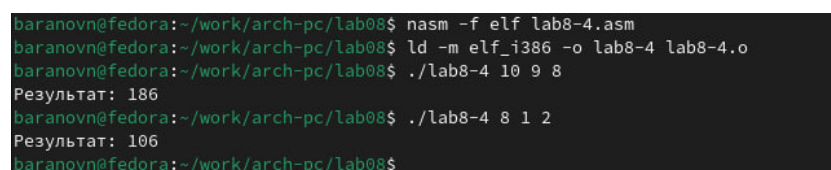
    add esi, ebx

    loop next

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

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

Рис. 4.1: Создаем файл и вписываем программу N18



```
baranovn@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
baranovn@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
baranovn@fedora:~/work/arch-pc/lab08$ ./lab8-4 10 9 8
Результат: 186
baranovn@fedora:~/work/arch-pc/lab08$ ./lab8-4 8 1 2
Результат: 106
baranovn@fedora:~/work/arch-pc/lab08$
```

Рис. 4.2: Создаем объектный файл и проверяем работу программы на нескольких примерах

Аналитическим методом я получил соответствующие ответы.

5 Выводы

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