

# **Лабораторная работа №8**

**Архитектура Компьютера**

Михальский Кирилл Алексеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Задания для самостоятельной работы</b>	<b>14</b>
<b>6</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

# Список иллюстраций

4.1	Название рисунка . . . . .	8
4.2	Название рисунка . . . . .	9
4.3	Название рисунка . . . . .	9
4.4	Название рисунка . . . . .	10
4.5	Название рисунка . . . . .	10
4.6	Название рисунка . . . . .	11
4.7	Название рисунка . . . . .	11
4.8	Название рисунка . . . . .	12
4.9	Название рисунка . . . . .	12
4.10	Название рисунка . . . . .	13
4.11	Название рисунка . . . . .	13
5.1	Название рисунка . . . . .	14

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

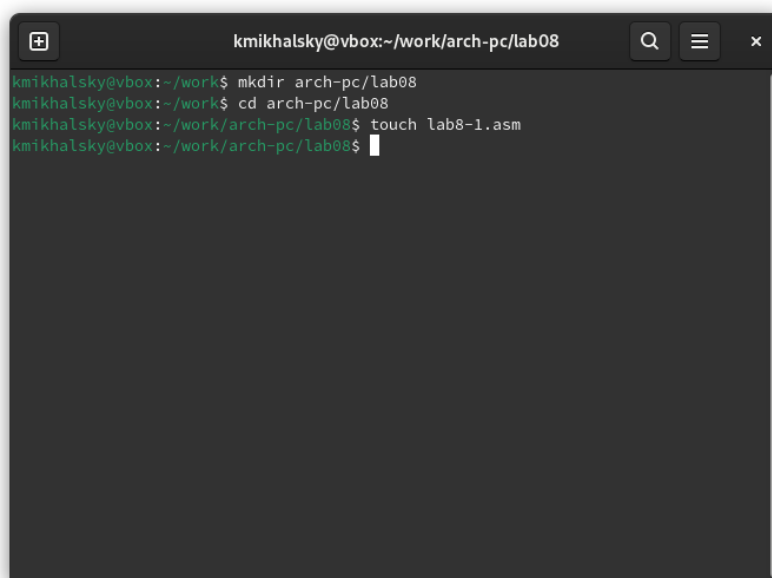
1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной работы

### 3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

## 4 Выполнение лабораторной работы

1. Создал каталог и файл lab8-1. Ввел текст программы.

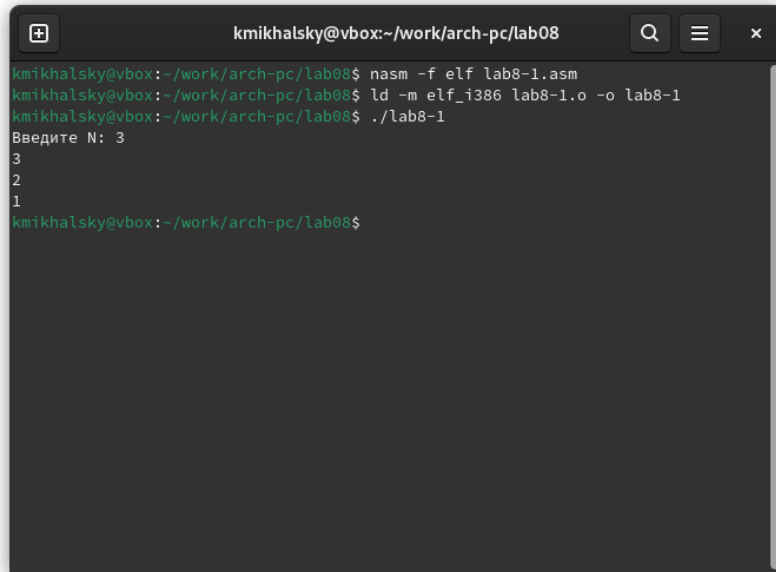
A terminal window with a dark background and light green text. The title bar at the top reads 'kmikhalsky@vbox:~/work/arch-pc/lab08'. The terminal shows the following commands and their outputs:

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

Рис. 4.1: Название рисунка

Проверил работу программы.



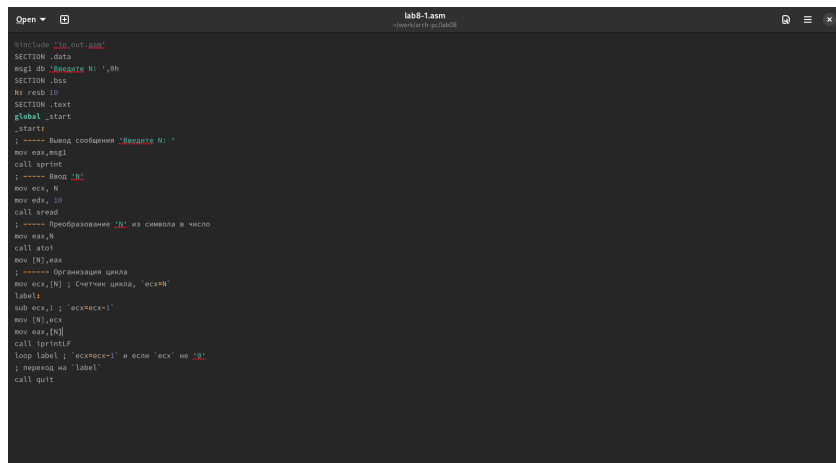


A terminal window titled 'kmikhalsky@vbox:~/work/arch-pc/lab08'. The user enters the following commands: `nasm -f elf lab8-1.asm`, `ld -m elf_i386 lab8-1.o -o lab8-1`, and `./lab8-1`. The program prompts 'Введите N: 3', the user enters '3', then '2', then '1'. The prompt appears again.

```
kmikhalsky@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
kmikhalsky@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 3
3
2
1
kmikhalsky@vbox:~/work/arch-pc/lab08$
```

Рис. 4.2: Название рисунка

Ввел необходимые изменения в работу программы.

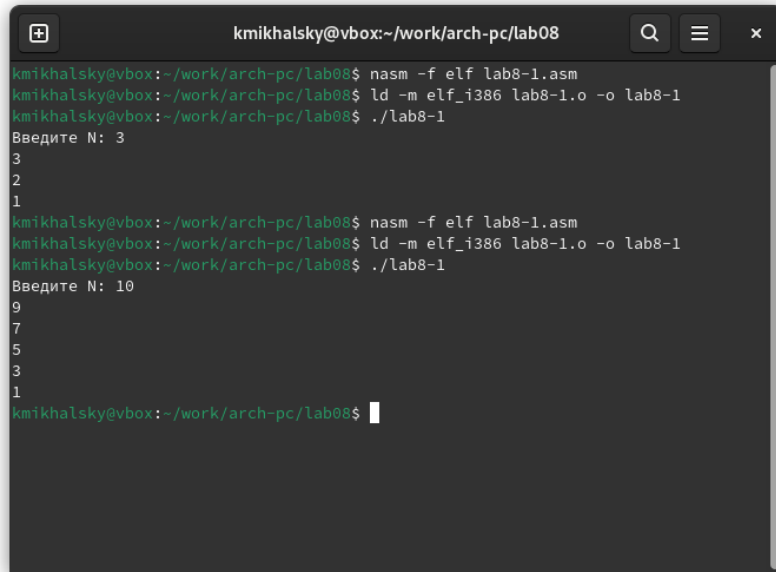


A screenshot of the 'lab8-1.asm' source code in a text editor. The code includes a header file, defines data and text sections, and implements a program that reads a number 'N' and prints a sequence of characters based on 'N'.

```
Open lab8-1.asm
#include "ig-out.asm"
SECTION .data
msg1 db "Введите N: ",0h
SECTION .bss
N resb 10
SECTION .text
global _start
_start:
; ----- Ввод сообщения "Введите N: "
mov eax,msg1
call sprintf
call sprint
; ----- Ввод "N"
mov ecx, N
mov edx, 10
call read
; ----- Преобразование "N" из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, "если N"
label1:
sub ecx,1 ; "если не 1"
mov [N],ecx
mov eax,[N]
call sprintf
call printf
loop label1 ; "если не 1" и если "если" не "1"
; переход на "label1"
call quit
```

Рис. 4.3: Название рисунка

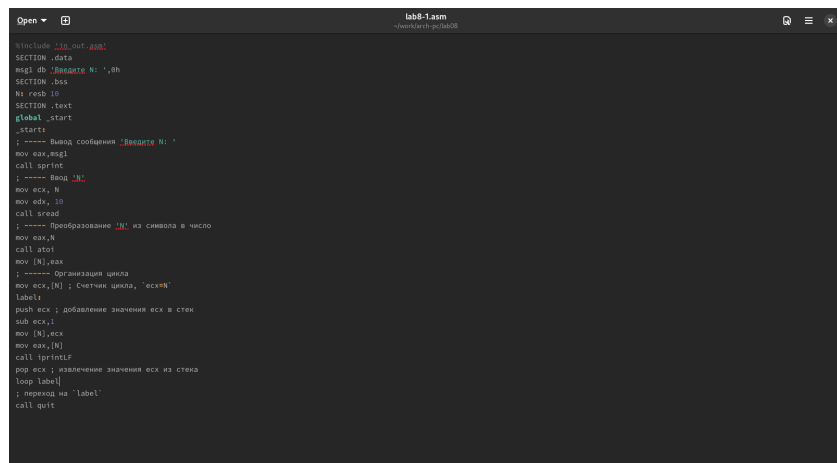
Проверил работу измененной программы.



A terminal window titled 'kmikhalsky@vbox:~/work/arch-pc/lab08'. The user enters the following commands: `nasm -f elf lab8-1.asm`, `ld -m elf_i386 lab8-1.o -o lab8-1`, and `./lab8-1`. The program prompts 'Введите N: 3' and the user enters '3'. The program then prints '3', '2', and '1'. The user enters another command `nasm -f elf lab8-1.asm`, followed by `ld -m elf_i386 lab8-1.o -o lab8-1` and `./lab8-1`. The program prompts 'Введите N: 10' and the user enters '10'. The program then prints '9', '7', '5', '3', and '1'. The terminal ends with the prompt `kmikhalsky@vbox:~/work/arch-pc/lab08$`.

Рис. 4.4: Название рисунка

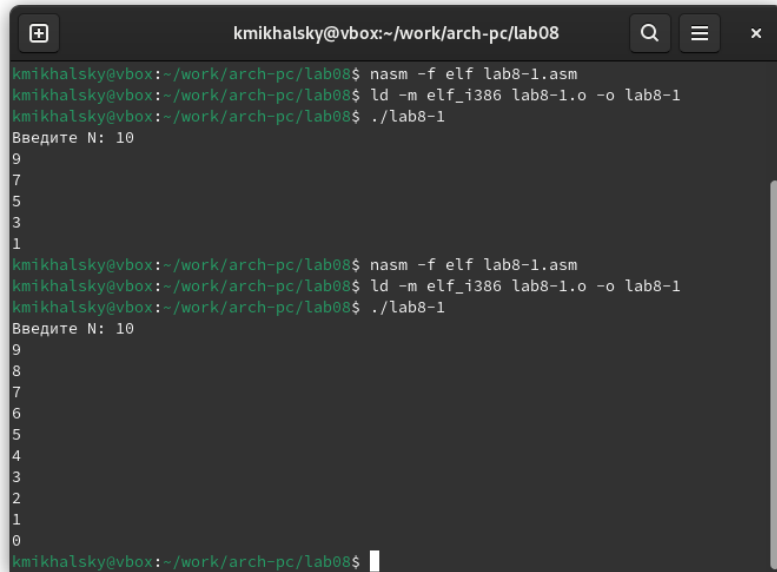
Ввел изменения в код программы.



A screenshot of an assembly source code file named 'lab8-1.asm'. The code includes a comment `include i386.def.asm` and defines sections for data, bss, and text. It uses macros `%Baudrate` and `%N` for baud rate and iteration count. The code implements a loop that reads a character, converts it to a number, and prints it. Comments in Russian describe the logic: 'Вывод сообщения "Baudrate N: "', 'Преобразование "N" из символа в число', 'Организация цикла', 'Счетчик цикла, "еск"', 'добавление значения еск в стек', and 'назначение значения еск из стека'. The code ends with `call quit`.

Рис. 4.5: Название рисунка

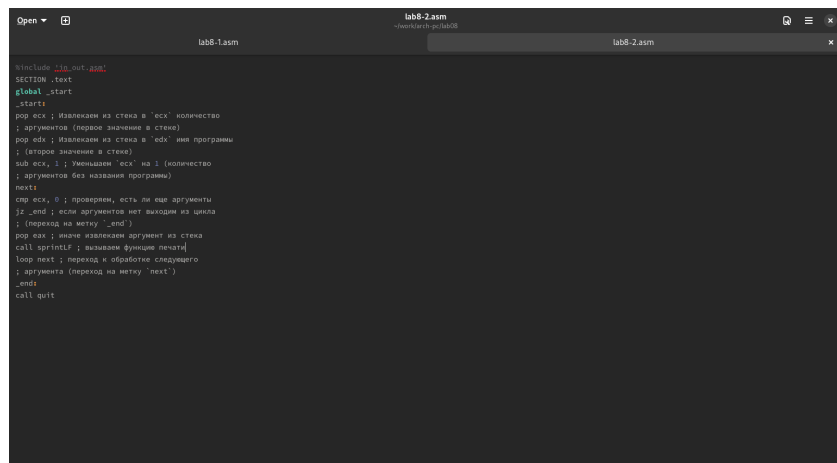
Проверил работу измененной программы.



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

Рис. 4.6: Название рисунка

Создал файл lab8-2 и ввел текст программы.

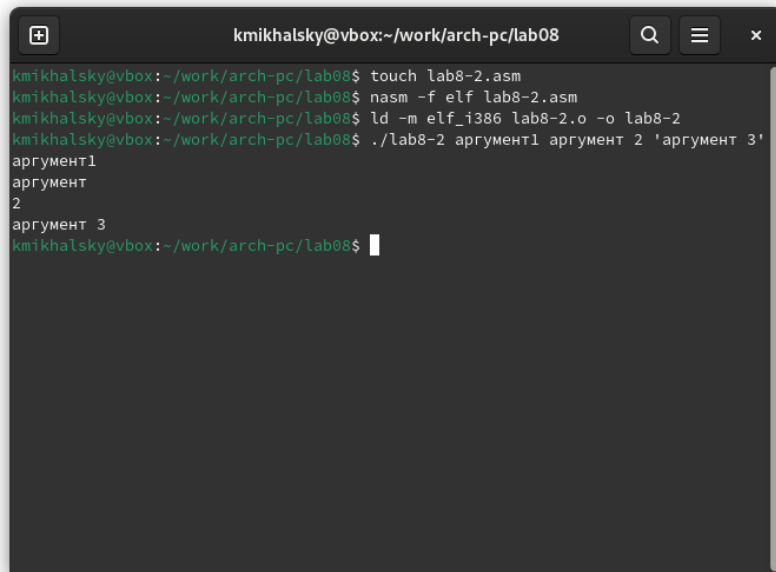


```
lab8-2.asm
~work/arch-pc/lab08

#include <stdio.h>
SECTION .text
global _start
_start:
pop ecx ; Инициализируем стек в 'ecx' количество
; аргументов (первое значение в стеке)
pop esi ; Инициализируем стек в 'esi' имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; Проверим, есть ли еще аргументы
jz _end ; Если аргументов нет выходим из цикла
; (переход на метку '_end')
pop eax ; Иначе инициализируем аргумент из стека
call sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку 'next')
_end:
call quit
```

Рис. 4.7: Название рисунка

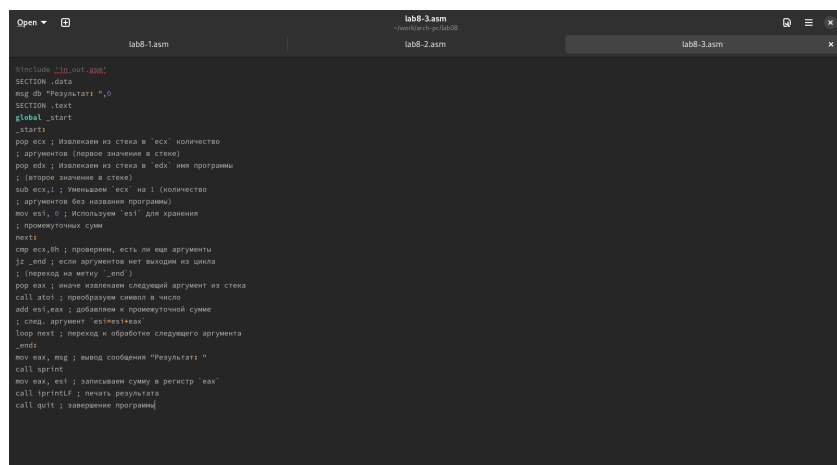
Запустил с необходимыми аргументами. Программа обрабатывает все введенные аргументы.



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

Рис. 4.8: Название рисунка

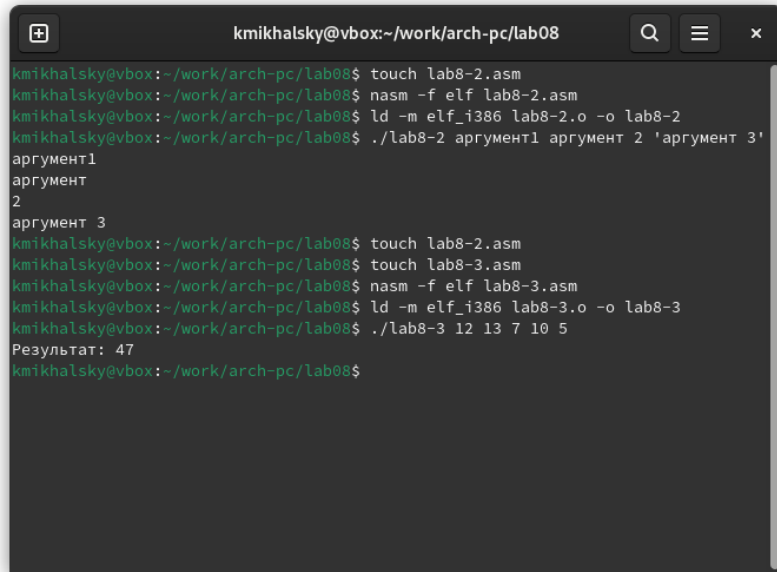
ВВе текст программы lab8-3.



```
lab8-3.asm
#include <stdio.h>
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 ; Проверим, есть ли еще аргументы
    jz _end ; Если аргументов нет, выходим из цикла
    ; (переход на метку '_end')
    pop eax ; Извлекаем следующий аргумент из стека
    call atoi ; Преобразуем символ в число
    add esi,eax ; Добавим к промежуточной сумме
    ; След. аргумент 'eax' к 'esi'
    loop next ; Переход к обработке следующего аргумента
_end:
    mov eax,msg ; Вывод сообщения "Результат: "
    call printf
    mov ecx,esi ; Записываем сумму в регистр 'eax'
    call iprintf ; Печатаем результат
    call quit ; Завершение программы
```

Рис. 4.9: Название рисунка

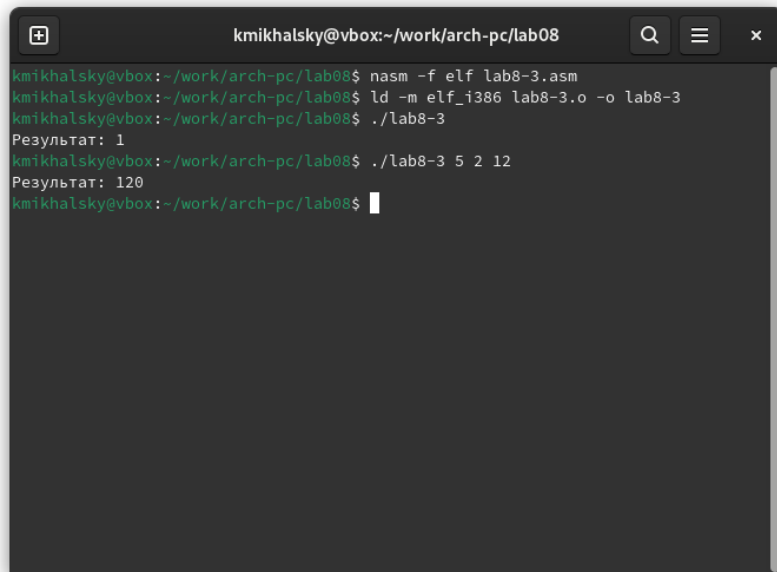
Проверил работу программы.



```
kmikhalsky@vbox:~/work/arch-pc/lab08$ touch lab8-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
kmikhalsky@vbox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
kmikhalsky@vbox:~/work/arch-pc/lab08$ touch lab8-2.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ touch lab8-3.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
kmikhalsky@vbox:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
kmikhalsky@vbox:~/work/arch-pc/lab08$
```

Рис. 4.10: Название рисунка

Внес необходимые изменения и проверил правильность работы программы:

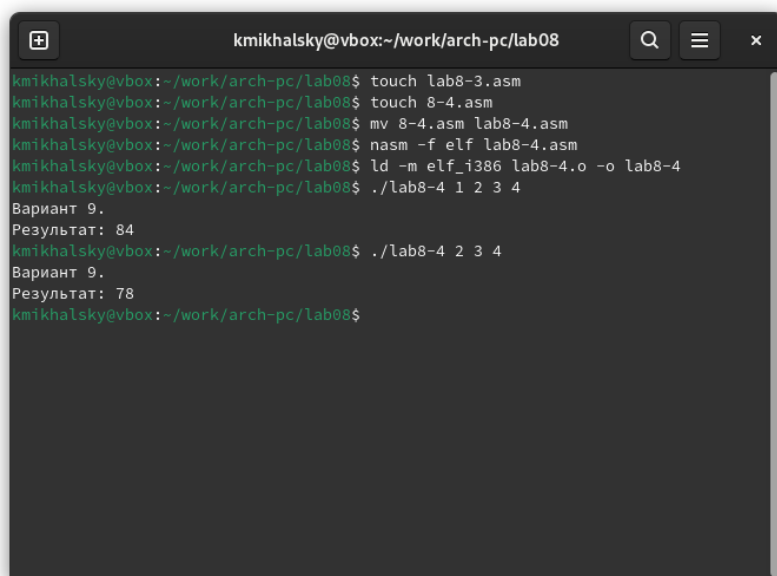


```
kmikhalsky@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
kmikhalsky@vbox:~/work/arch-pc/lab08$ ./lab8-3
Результат: 1
kmikhalsky@vbox:~/work/arch-pc/lab08$ ./lab8-3 5 2 12
Результат: 120
kmikhalsky@vbox:~/work/arch-pc/lab08$
```

Рис. 4.11: Название рисунка

## 5 Задания для самостоятельной работы

1. Написал программу по условию. Проверил правильность работы программы.



```
kmikhalsky@vbox:~/work/arch-pc/lab08$ touch lab8-3.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ touch 8-4.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ mv 8-4.asm lab8-4.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
kmikhalsky@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
kmikhalsky@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Вариант 9.
Результат: 84
kmikhalsky@vbox:~/work/arch-pc/lab08$ ./lab8-4 2 3 4
Вариант 9.
Результат: 78
kmikhalsky@vbox:~/work/arch-pc/lab08$
```

Рис. 5.1: Название рисунка

## **6 Выводы**

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

## **Список литературы**