

# Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера  
Хан Георгий Игоревич

# Оглавление

1 Цель работы .....	2
2 Задание .....	2
3 Теоретическое введение .....	2
4 Выполнение лабораторной работы .....	3
4.1 Реализация циклов в NASM .....	3
4.2 Обработка аргументов командной строки .....	7
4.3 Задание для самостоятельной работы .....	9
5 Выводы.....	10
6 Список литературы.....	11

## 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

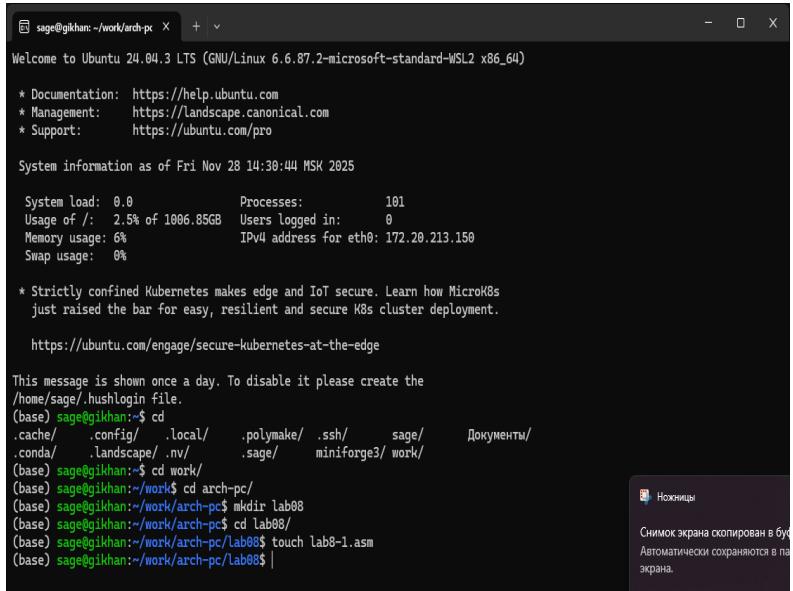
## 3 Теоретическое введение

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

## 4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 (рис. 1).



```
sage@gikhan:~/work/arch-pc X + v
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.6.87.2-microsoft-standard-WSL2 x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Fri Nov 28 14:30:44 MSK 2025

System load: 0.0      Processes: 101
Usage of /: 2.5% of 1006.85GB  Users logged in: 0
Memory usage: 6%
Swap usage: 0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.

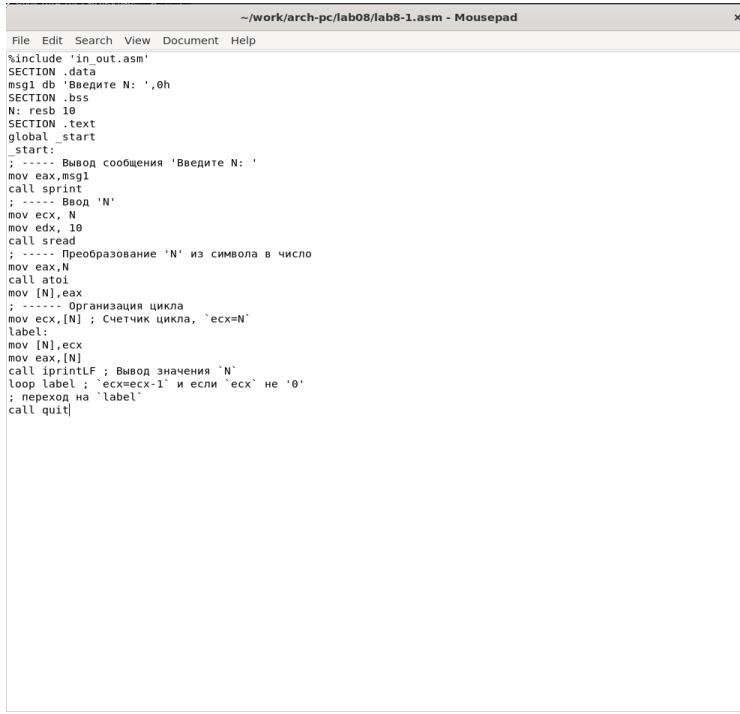
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/sage/.hushlogin file.

(base) sage@gikhan:~$ cd
(base) sage@gikhan:~$ .cache/ .config/ .local/ .polymake/ .ssh/ sage/ Документы/
(base) sage@gikhan:~$ cd work/
(base) sage@gikhan:~/work$ cd arch-pc/
(base) sage@gikhan:~/work/arch-pc$ mkdir lab08
(base) sage@gikhan:~/work/arch-pc$ cd lab08/
(base) sage@gikhan:~/work/arch-pc/lab08$ touch lab8-1.asm
(base) sage@gikhan:~/work/arch-pc/lab08$ |
```

Рис. 1: Создание каталога

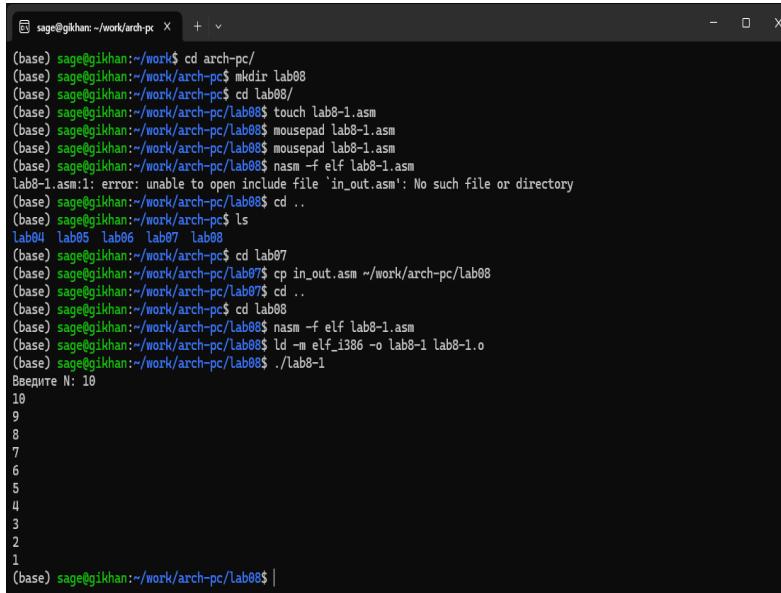
Копирую в созданный файл программу из листинга. (рис. 2).



```
~/work/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
%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
label:
mov ecx,[N] ; Счетчик цикла, `ecx=N'
mov eax,[N]
call printfLF ; Вывод значения 'N'
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit|
```

Рис. 2: Копирование программы из листинга

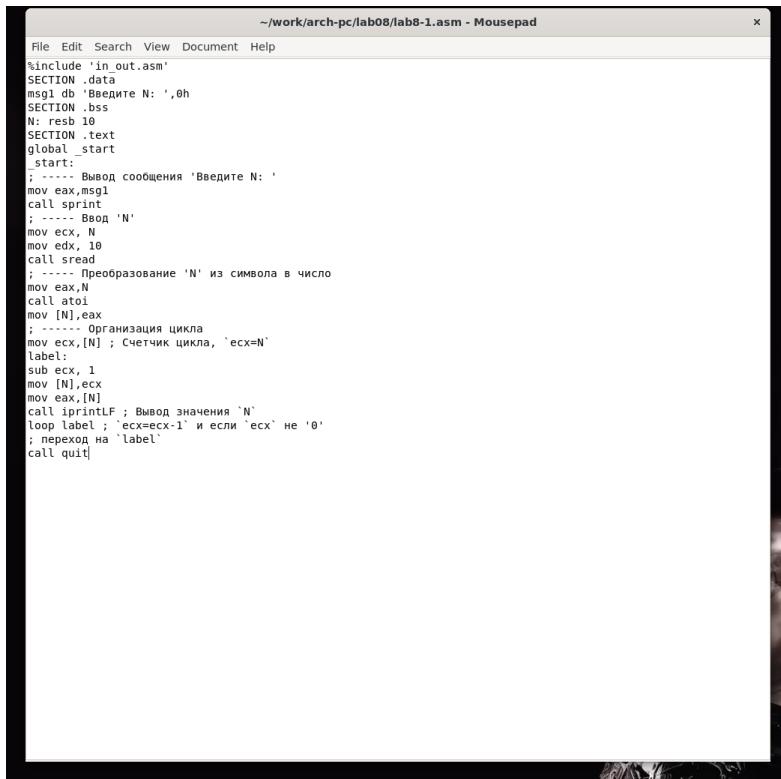
Запускаю программу, она показывает работу циклов в NASM (рис. 3).



```
(base) sage@gikhan:~/work/arch-pc$ cd arch-pc/
(base) sage@gikhan:~/work/arch-pc$ mkdir lab08
(base) sage@gikhan:~/work/arch-pc$ cd lab08/
(base) sage@gikhan:~/work/arch-pc/Lab08$ touch lab8-1.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-1.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-1.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:1: error: unable to open include file 'in_out.asm': No such file or directory
(base) sage@gikhan:~/work/arch-pc/Lab08$ cd ..
(base) sage@gikhan:~/work/arch-pc$ ls
lab05 lab06 lab07 lab08
(base) sage@gikhan:~/work/arch-pc$ cd lab07
(base) sage@gikhan:~/work/arch-pc/lab07$ cp in_out.asm ~/work/arch-pc/Lab08
(base) sage@gikhan:~/work/arch-pc/Lab07$ cd ..
(base) sage@gikhan:~/work/arch-pc$ cd lab08
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-1.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
(base) sage@gikhan:~/work/arch-pc/Lab08$ |
```

Рис. 3: Запуск программы

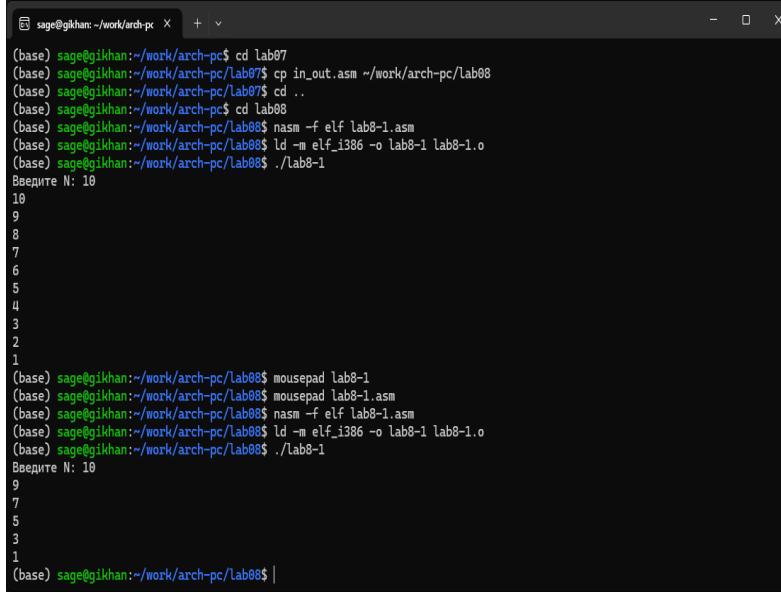
Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx (рис. 4).



```
~/work/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
%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 read
; ---- 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
mov [N],ecx
mov eax,[N]
call printf ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
; call quit|
```

Рис. 4: Изменение программы

Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. 5).



The screenshot shows a terminal window with the following session:

```
sage@gikhan:~/work/arch-pc$ cd lab07
(base) sage@gikhan:~/work/arch-pc$ cp in_out.asm ~/work/arch-pc/lab08
(base) sage@gikhan:~/work/arch-pc/lab07$ cd ..
(base) sage@gikhan:~/work/arch-pc$ cd lab08
(base) sage@gikhan:~/work/arch-pc$ nasm -f elf lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) sage@gikhan:~/work/arch-pc$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-1
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ nasm -f elf lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) sage@gikhan:~/work/arch-pc$ ./Lab8-1
Введите N: 10
9
7
5
3
1
(base) sage@gikhan:~/work/arch-pc$ |
```

Рис. 5: Запуск измененной программы

Добавляю команды push и pop в программу (рис. 6).

```
%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
sub ecx, 1
mov [N],ecx
mov eax,[N]
call printf ; Вывод значения `N`
pop ecx
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Рис. 6: Добавление push и pop в цикл программы

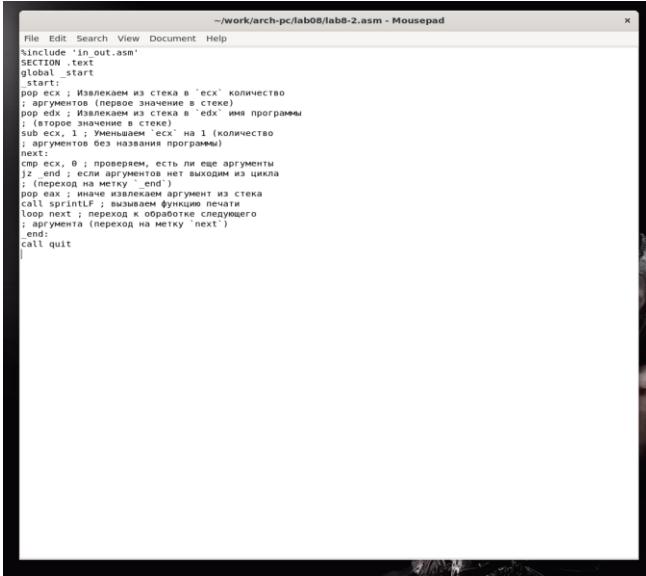
Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. 7).

```
3
2
1
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ nasm -f elf lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) sage@gikhan:~/work/arch-pc$ ./lab8-1
Введите N: 10
9
7
5
3
1
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ nasm -f elf lab8-1.asm
(base) sage@gikhan:~/work/arch-pc$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) sage@gikhan:~/work/arch-pc$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
(base) sage@gikhan:~/work/arch-pc$ |
```

Рис. 7: Запуск измененной программы

## 4.2 Обработка аргументов командной строки

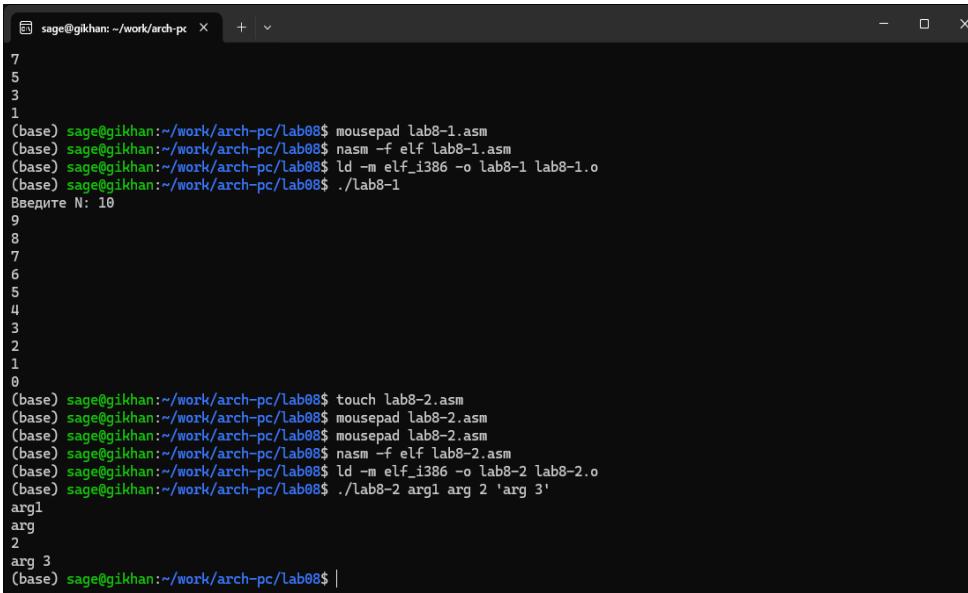
Создаю новый файл для программы и копирую в него код из следующего листинга (рис. 8).



```
~/work/arch-pc/lab08/lab8-2.asm - Mousepad
File Edit Search View Document Help
#include 'in_out.asm'
DATETIME .text
global start
start:
    pop ecx ; Извлекаем из стека в 'ecx' количество
    jz end ; если (последнее 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, переданных называемой программе)
next:
    cmp ecx, 0 ; проверяю, есть ли еще аргументы
    jz end ; если аргументов нет выходим из цикла
    ; (переход на метку 'end')
    pop eax ; иначе извлекаем аргумент из стека
    ; с помощью команды 'pop' и выводим его на экран
    loop next ; переход к обработке следующего
    ; аргумента (переход на метку 'next')
end:
call quit
```

Рис. 8: Копирование программы из листинга

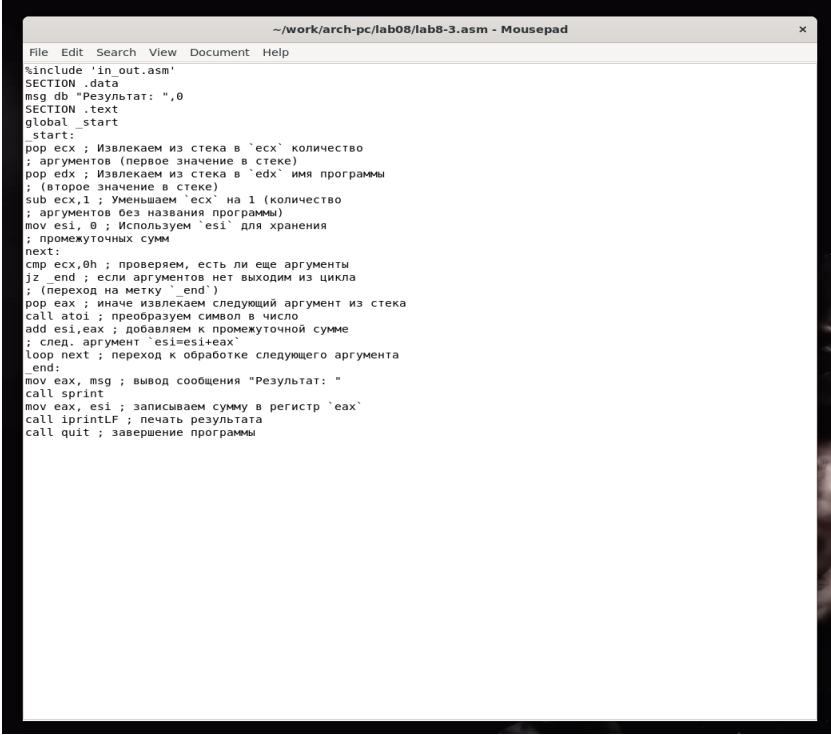
Компилирую программу и запускаю, указав аргументы. Программой было возвращено то же количество аргументов, что и было введено (рис. 9).



```
 sage@gikhan:~/work/arch-pc ~ x + v
7
5
3
1
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-1.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-1.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./Lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
(base) sage@gikhan:~/work/arch-pc/Lab08$ touch lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
(base) sage@gikhan:~/work/arch-pc/Lab08$ |
```

Рис. 9: Запуск второй программы

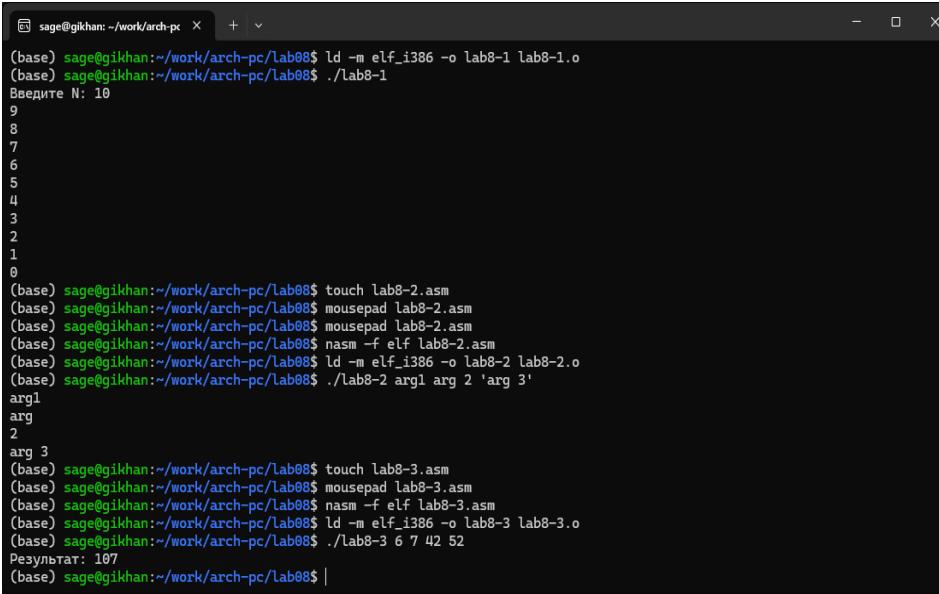
Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 10).



```
~/work/arch-pc/lab08/lab8-3.asm - Mousepad
File Edit Search View Document Help
%include 'in.out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    ; Извлекаем из стека в `есх` количество
    ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
    ; (второе значение в стеке)
    sub ecx,1 ; Уменьшаем `есх` 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`)
    ; перед ехах значение в есх, получаем следующий аргумент из стека
    call atoi ; преобразуем символ в число
    add esi,eah ; добавляем к промежуточной сумме
    ; след. аргумент `esi=esi+eah`
    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 iprintfLF ; печать результата
    call quit ; завершение программы
```

Рис. 10: Копирование программы из третьего листинга

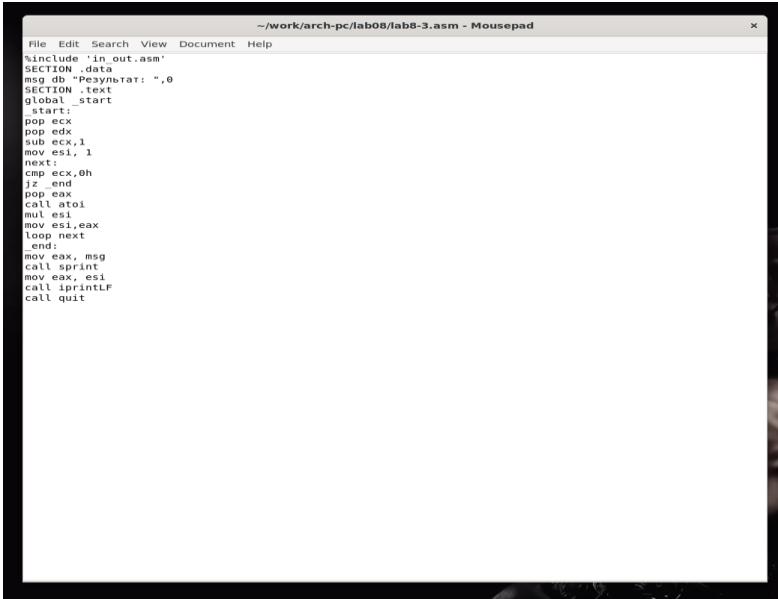
Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 11).



```
sage@gikhan:~/work/arch-pc x + v
(base) sage@gikhan:~/work/arch-pc$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) sage@gikhan:~/work/arch-pc$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
(base) sage@gikhan:~/work/arch-pc$ touch lab8-2.asm
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-2.asm
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-2.asm
(base) sage@gikhan:~/work/arch-pc$ nasm -f elf lab8-2.asm
(base) sage@gikhan:~/work/arch-pc$ ld -m elf_i386 -o lab8-2 lab8-2.o
(base) sage@gikhan:~/work/arch-pc$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
(base) sage@gikhan:~/work/arch-pc$ touch lab8-3.asm
(base) sage@gikhan:~/work/arch-pc$ mousepad lab8-3.asm
(base) sage@gikhan:~/work/arch-pc$ nasm -f elf lab8-3.asm
(base) sage@gikhan:~/work/arch-pc$ ld -m elf_i386 -o lab8-3 lab8-3.o
(base) sage@gikhan:~/work/arch-pc$ ./Lab8-3 6 7 42 52
Результат: 107
(base) sage@gikhan:~/work/arch-pc$ |
```

Рис. 11: Запуск третьей программы

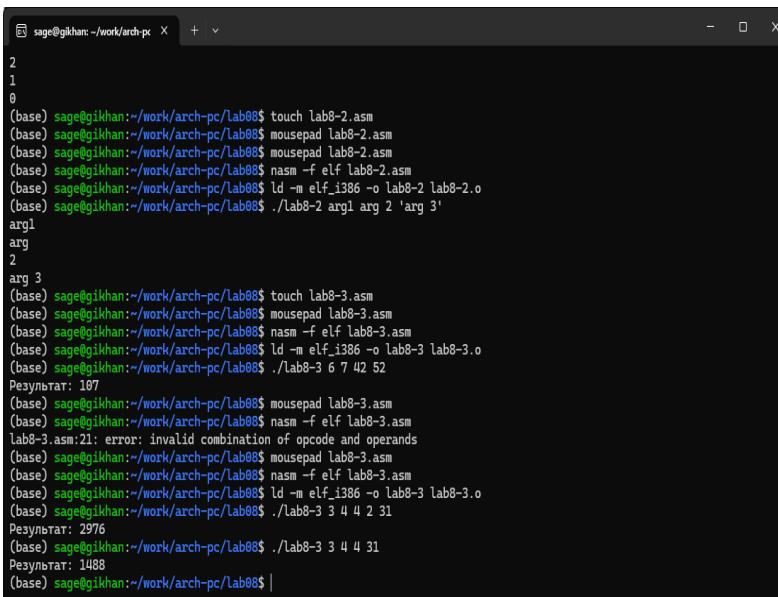
Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 12).



```
~/work/arch-pc/lab08/lab8-3.asm - Mousepad
File Edit Search View Document Help
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    push rbp
    pop rbp
    mov rsi,1
    mov rdi,esi
    next:
    cmp rdx,0h
    jz end
    pop rax
    call atoi
    mul rsi
    mov rsi,rax
    loop next
end:
    mov rax, msg
    call sprint
    mov rax,esi
    call iprintf
    call quit
```

Рис. 12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 13).



```
sage@gikhan:~/work/arch-pc > + <
2
1
8
(base) sage@gikhan:~/work/arch-pc/Lab08$ touch lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-2.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./Lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
(base) sage@gikhan:~/work/arch-pc/Lab08$ touch lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./Lab8-3 6 7 42 52
Результат: 107
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-3.asm
lab8-3.asm:21: error: invalid combination of opcode and operands
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./Lab8-3 3 4 2 31
Результат: 2976
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./lab8-3 3 4 31
Результат: 1488
(base) sage@gikhan:~/work/arch-pc/Lab08$ |
```

Рис. 13: Запуск измененной третьей программы

### 4.3 Задание для самостоятельной работы

Пишу программу, которая будет находить сумму значений для функции  $f(x) = 10x - 4$ , которая совпадает с моим девятым вариантом (рис. 14).

```
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 10x - 4", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_func
    call sprintLF
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0

next:
    cmp esi, 0h
    jne end
    pop eax
    call atoi
    mov ebx, 10
    mul ebx
    sub eax, 4
    add esi, eax
    loop next

end:
    mov eax, msg_result
    call sprint
    mov eax, esi
    call iprintLF
    call quit
```

Рис. 14: Написание программы для самостоятельной работы

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. 15).

```
sage@gikhan:~/work/arch-pc: ~ + v
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./Lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
(base) sage@gikhan:~/work/arch-pc/Lab08$ touch lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./lab8-3 6 7 42 52
Результат: 107
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-3.asm
lab8-3.asm:21: error: invalid combination of opcode and operands
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-3.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./lab8-3 6 7 42 52
Результат: 2976
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./Lab8-3 3 4 4 31
Результат: 1488
(base) sage@gikhan:~/work/arch-pc/Lab08$ touch lab8-4.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ mousepad lab8-4.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ nasm -f elf lab8-4.asm
(base) sage@gikhan:~/work/arch-pc/Lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
(base) sage@gikhan:~/work/arch-pc/Lab08$ ./Lab8-4 1 2 3
Функция: f(x) = 10x - 4
Результат: 48
(base) sage@gikhan:~/work/arch-pc/Lab08$ |
```

Рис. 15: Запуск программы для самостоятельной работы

## 5 Выводы

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

## 6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №8
3. Программирование на языке ассемблера NASM Столяров А. В.