

# **Отчёт по лабораторной работе №8**

**Дисциплина: Архитектура компьютера**

Бызова Мария Олеговна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация циклов в NASM . . . . .	6
<b>3</b>	<b>Выполнение заданий для самостоятельной работы</b>	<b>14</b>
<b>4</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

2.1	Создание необходимой директории и файла . . . . .	6
2.2	Редактирование файла . . . . .	7
2.3	Создание исполняемого файла . . . . .	7
2.4	Запуск исполняемого файла . . . . .	7
2.5	Редактирование файла . . . . .	8
2.6	Создание исполняемого файла . . . . .	8
2.7	Запуск исполняемого файла . . . . .	8
2.8	Запуск исполняемого файла . . . . .	9
2.9	Редактирование файла . . . . .	10
2.10	Создание исполняемого файла . . . . .	10
2.11	Запуск исполняемого файла . . . . .	10
2.12	Создание файла . . . . .	11
2.13	Редактирование файла . . . . .	11
2.14	Создание исполняемого файла . . . . .	11
2.15	Запуск исполняемого файла . . . . .	11
2.16	Создание файла . . . . .	12
2.17	Редактирование файла . . . . .	12
2.18	Создание исполняемого файла . . . . .	12
2.19	Запуск исполняемого файла . . . . .	12
2.20	Редактирование файла . . . . .	13
2.21	Создание исполняемого файла . . . . .	13
2.22	Запуск исполняемого файла . . . . .	13
3.1	Создание файла . . . . .	14
3.2	Написание программы . . . . .	15
3.3	Создание исполняемого файла . . . . .	15
3.4	Запуск исполняемого файла . . . . .	15
3.5	Запуск исполняемого файла . . . . .	15

## Список таблиц

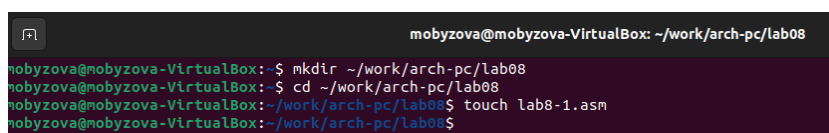
# 1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

## 2 Выполнение лабораторной работы

### 2.1 Реализация циклов в NASM

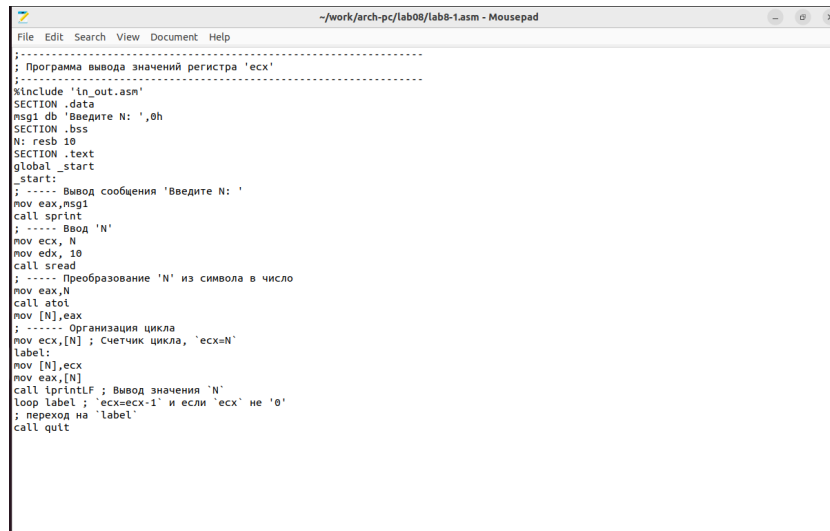
С помощью утилиты `mkdir` создаем директорию, в которой будем создавать файлы с программами для лабораторной работы №8. Переходим в созданный каталог с помощью утилиты `cd`. С помощью утилиты `touch` создаем файл `lab8-1.asm` (рис. [2.1]).



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

Рис. 2.1: Создание необходимой директории и файла

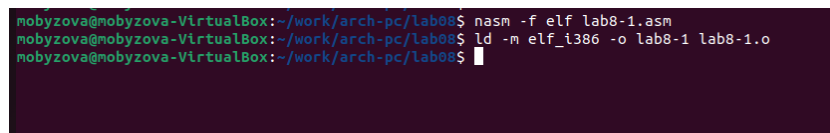
При помощи Midnight Commander открываем созданный файл `lab8-1.asm`, вставляем в него программу вывода значений регистра `ecx` из листинга 8.1 (рис. [2.2]).



```
File Edit Search View Document Help
;-----
; Программа вывода значений регистра 'ecx'
;-----
%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:
mov [N],ecx
mov eax,[N]
call tprintf ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit
```

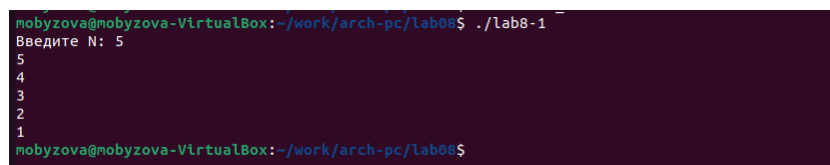
Рис. 2.2: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.3], [2.4]).



```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$
```

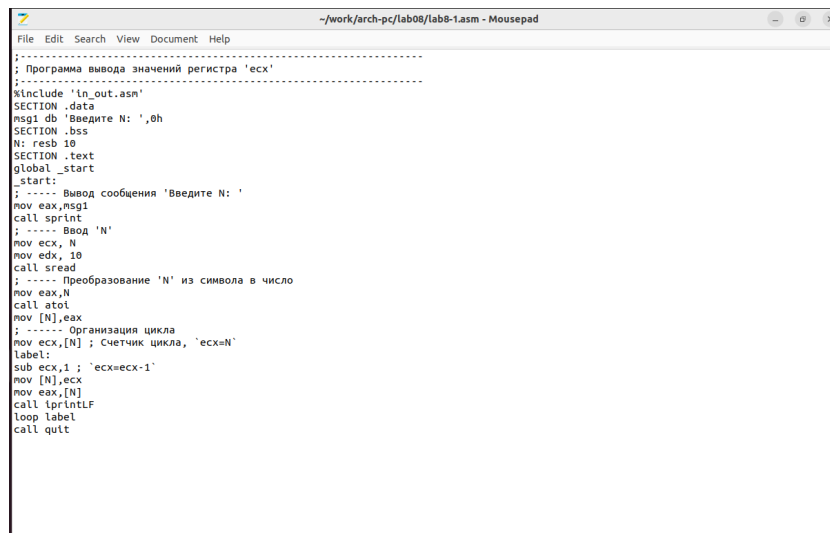
Рис. 2.3: Создание исполняемого файла



```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.4: Запуск исполняемого файла

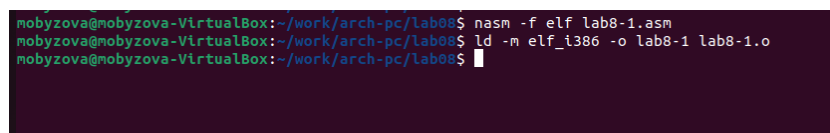
Далее мы изменяем текст программы, добавив изменение значение регистра `ecx` в цикле (рис. [2.5]).



```
File Edit Search View Document Help
;-----
; Программа вывода значений регистра 'ecx'
;-----
%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 lprintLF
loop label
call quit
```

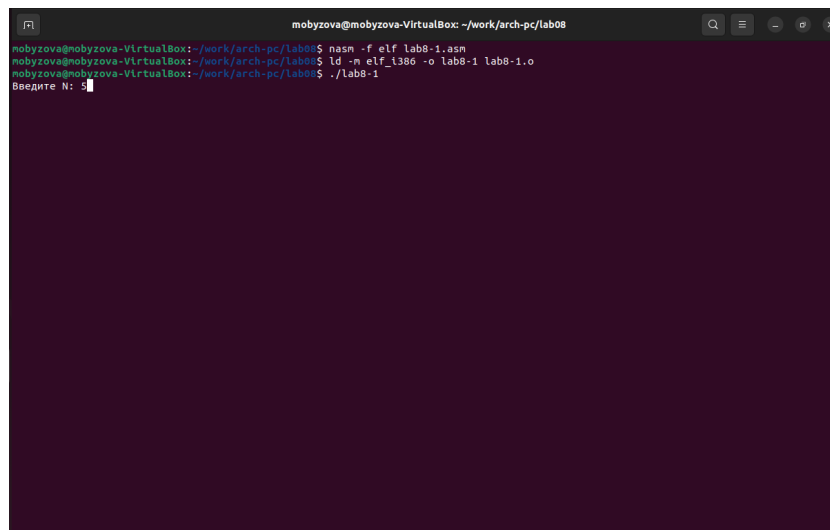
Рис. 2.5: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.6], [2.7], [2.8]).



```
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$
```

Рис. 2.6: Создание исполняемого файла



```
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
```

Рис. 2.7: Запуск исполняемого файла



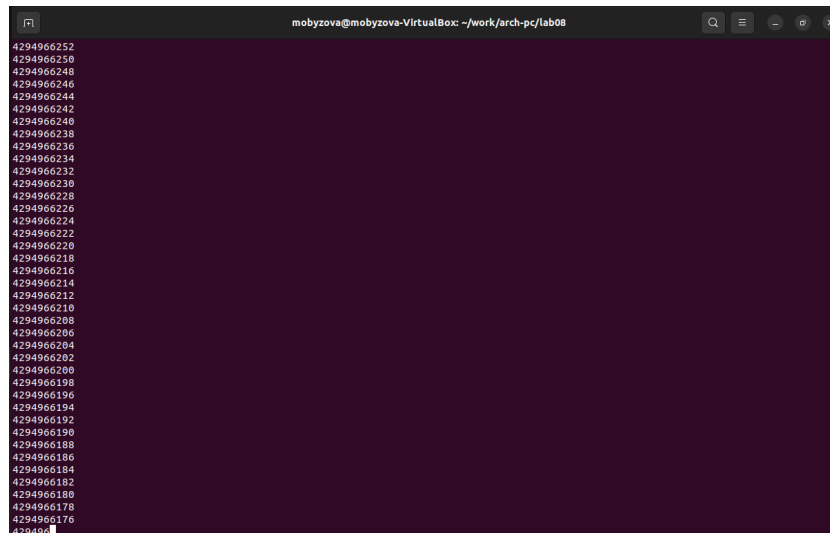
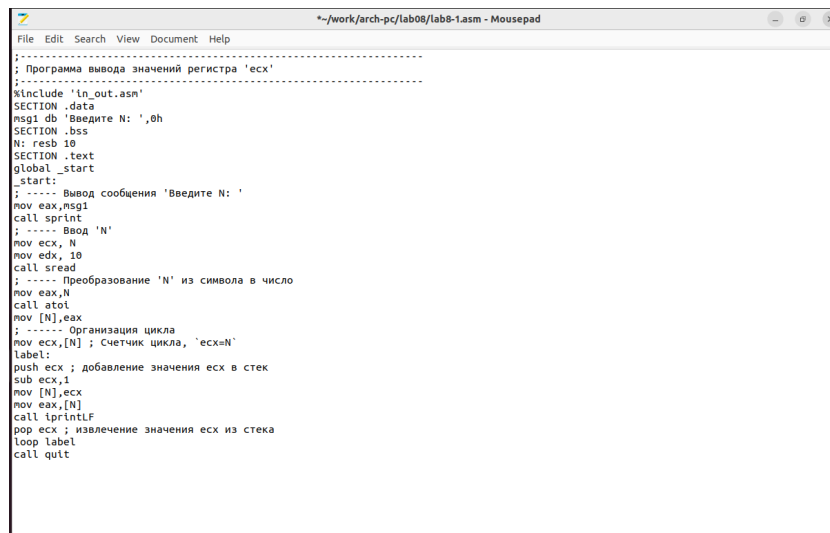


Рис. 2.8: Запуск исполняемого файла

Таким образом, использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Регистр `ecx` принимает в цикле некорректные значения, а число проходов цикла не соответствует значению `N` введенному с клавиатуры.

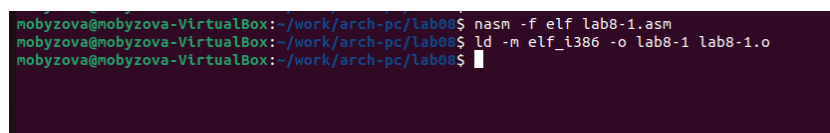
Затем мы внесём изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop` (рис. [2.9]).



```
File Edit Search View Document Help
;-----
; Программа вывода значений регистра 'ecx'
;-----
%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:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call tprintf
pop ecx ; извлечение значения ecx из стека
loop label
call quit
```

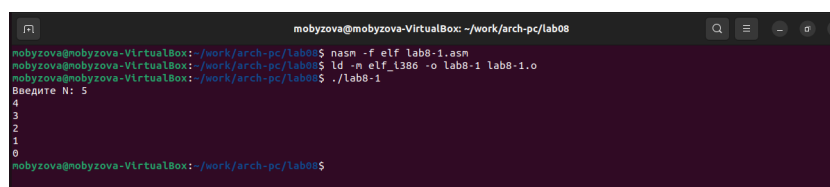
Рис. 2.9: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.10], [2.11])



```
mobyzoa@mobyzoa-VirtualBox: ~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mobyzoa@mobyzoa-VirtualBox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mobyzoa@mobyzoa-VirtualBox: ~/work/arch-pc/lab08$
```

Рис. 2.10: Создание исполняемого файла



```
mobyzoa@mobyzoa-VirtualBox: ~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mobyzoa@mobyzoa-VirtualBox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mobyzoa@mobyzoa-VirtualBox: ~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
mobyzoa@mobyzoa-VirtualBox: ~/work/arch-pc/lab08$
```

Рис. 2.11: Запуск исполняемого файла


В данном случае число проходов цикла соответствуют значению N введенному с клавиатуры, однако выводимые на экран значения будут принимать вид от “4” до “0” соответственно.

Создаем файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. [2.12]).

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ touch lab8-2.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.12: Создание файла

Внимательно изучив текст программы вычисления суммы аргументов командной строки из листинга 8.2, введем его в lab8-2.asm (рис. [2.13]).



```
File Edit Search View Document Help
*/work/arch-pc/lab08/lab8-2.asm - Mousepad
#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. 2.13: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.14], [2.15])

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
```

Рис. 2.14: Создание исполняемого файла

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 "аргумент 3"
аргумент1
аргумент
2
аргумент 3
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.15: Запуск исполняемого файла

Программа обрабатывает и выводит на экран 4 аргумента.

Создаем файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. [2.16]).

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

Рис. 2.16: Создание файла

Внимательно изучив текст программы вычисления суммы аргументов командной строки из листинга 8.3, введем его в lab8-3.asm (рис. [2.17]).

```
File Edit Search View Document Help
%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 sprintf
    mov eax,esi ; записываем сумму в регистр 'eax'
    call printf ; печать результата
    call quit ; завершение программы
```

Рис. 2.17: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.18], [2.19])

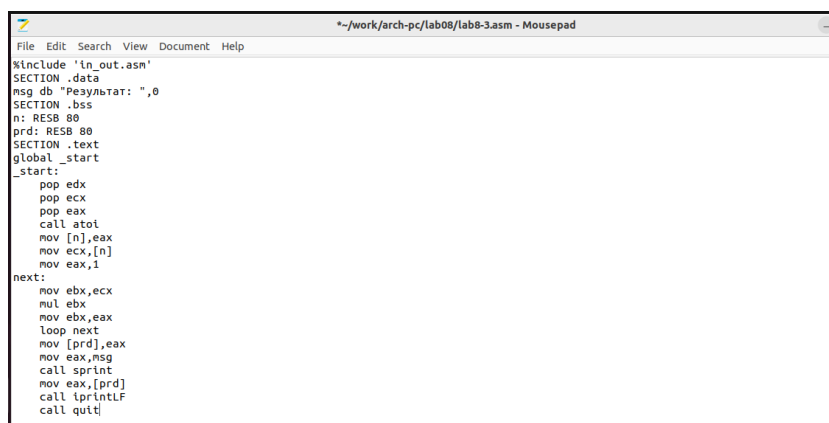
```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
```

Рис. 2.18: Создание исполняемого файла

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.19: Запуск исполняемого файла

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. [2.20]).

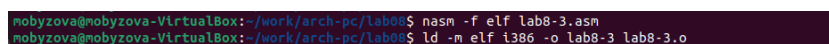


```
File Edit Search View Document Help
*~/work/arch-pc/lab08/lab8-3.asm - Mousepad

%include "in_out.asm"
SECTION .data
msg db "Результат: ",0
SECTION .bss
n: RESB 80
prd: RESB 80
SECTION .text
global _start
_start:
    pop edx
    pop ecx
    pop eax
    call atoi
    mov [n],eax
    mov ecx,[n]
    mov eax,1
next:
    mov ebx,ecx
    mul ebx
    mov ebx,eax
    loop next
    mov [prd],eax
    mov eax,msg
    call sprint
    mov eax,[prd]
    call iprintlnLF
    call quit
```

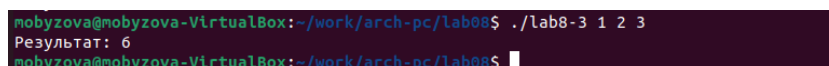
Рис. 2.20: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.21], [2.22])



```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
```

Рис. 2.21: Создание исполняемого файла



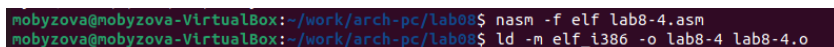
```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 1 2 3
Результат: 6
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.22: Запуск исполняемого файла

### 3 Выполнение заданий для самостоятельной работы

Задание: 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  $x = x_1, x_2, \dots, x_n$ .

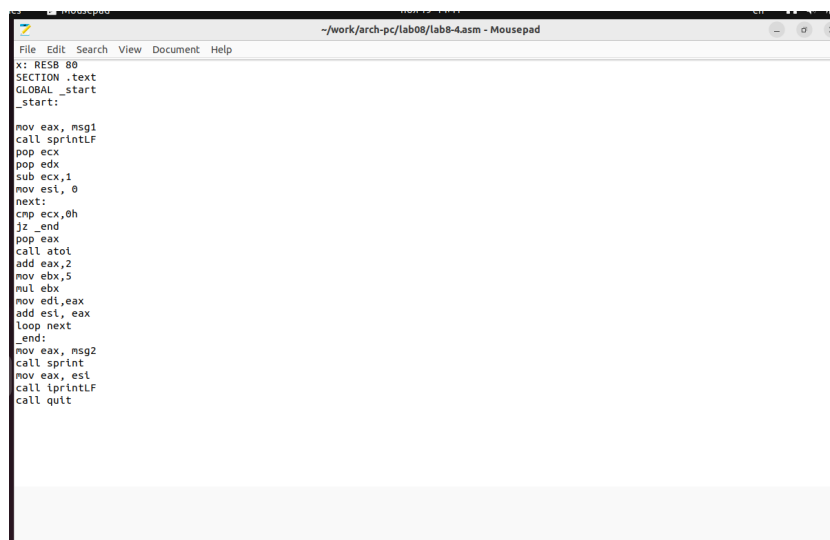
Создаем файл lab8-4.asm с помощью утилиты touch (рис. [3.1]).



```
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
mobyzova@mobyzova-VirtualBox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
```

Рис. 3.1: Создание файла

Открываем созданный файл для редактирования, вводим в него текст программы (рис. [3.2]).



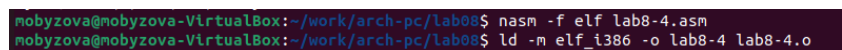
```
File Edit Search View Document Help
~/work/arch-pc/lab08/lab8-4.asm - Mousepad

x: RESB 80
SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprintf
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
add eax, 2
mov ebx, 5
mul ebx
mov edi, eax
add esi, eax
loop next
_end:
mov eax, msg2
call sprintf
mov eax, esi
call sprintf
call quit
```

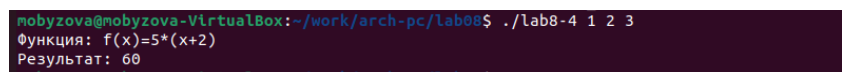
Рис. 3.2: Написание программы

Далее создаем исполняемый файл и запускаем его (рис. [3.3], [3.4], [3.5]).



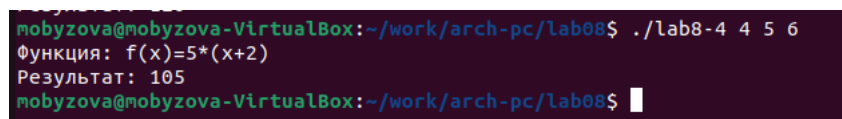
```
mobyzoza@mobyzoza-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
mobyzoza@mobyzoza-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
```

Рис. 3.3: Создание исполняемого файла



```
mobyzoza@mobyzoza-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция: f(x)=5*(x+2)
Результат: 60
```

Рис. 3.4: Запуск исполняемого файла



```
mobyzoza@mobyzoza-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 4 5 6
Функция: f(x)=5*(x+2)
Результат: 105
mobyzoza@mobyzoza-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 3.5: Запуск исполняемого файла

Посчитав аналитически сумму аргументов функции №10 из моего варианта, мы можем проверить правильность работы программы.

### Листинг №1. Изменённая программа lab8-3.asm

```

%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 1
next:
cmp ecx,0h
jz _end
pop eax
call atoi
imul esi, eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit

```

## Листинг №2. Задание для самостоятельной работы

```

%include 'in_out.asm'

SECTION .data
msg1: DB 'Функция: f(x)=5*(x+2) ',0
msg2: DB 'Результат: ',0

SECTION .bss

```



```

x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg1
call sprintLF
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
add eax, 2
mov ebx, 5
mul ebx
mov edi, eax
add esi, eax
loop next
_end:
mov eax, msg2
call sprint
mov eax, esi
call iprintLF
call quit

```

## 4 Выводы

В ходе лабораторной работы мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.