

Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютера

Маваси Башар

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Самостоятельная работа	15
4	Вывод	19

Список иллюстраций

2.1	Создание директории	5
2.2	Редактирование файла	6
2.3	Запуск исполняемого файла	7
2.4	Уменьшение индекса	7
2.5	Запуск исполняемого файла	8
2.6	Редактирование программы	8
2.7	Создание исполняемого файла	9
2.8	Создание файла	9
2.9	Вставляю текст в файл	10
2.10	Запуск исполняемого файла	10
2.11	Создание файла	11
2.12	вставляю программу	12
2.13	Запуск программы	12
2.14	Редактирование файла	13
2.15	Запуск программы	13
3.1	Создание файла	15
3.2	Редактирование файла	16
3.3	Запуск исполняемого файла	17

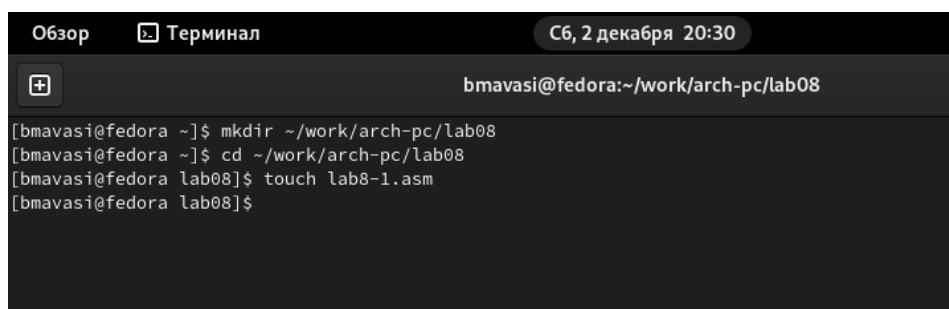
1 Цель работы

Получение навыков по организации циклов и работе со стеком на языке NASM.

2 Выполнение лабораторной работы

Шаг 1

С помощью утилиты `mkdir` создаю директорию `lab08`, перехожу в нее и создаю файл для работы. (рис. [2.1])



```
Обзор  Терминал  С6, 2 декабря 20:30
bmavasi@fedora:~/work/arch-pc/lab08
[bmavasi@fedora ~]$ mkdir ~/work/arch-pc/lab08
[bmavasi@fedora ~]$ cd ~/work/arch-pc/lab08
[bmavasi@fedora lab08]$ touch lab8-1.asm
[bmavasi@fedora lab08]$
```

Рис. 2.1: Создание директории

Шаг 2

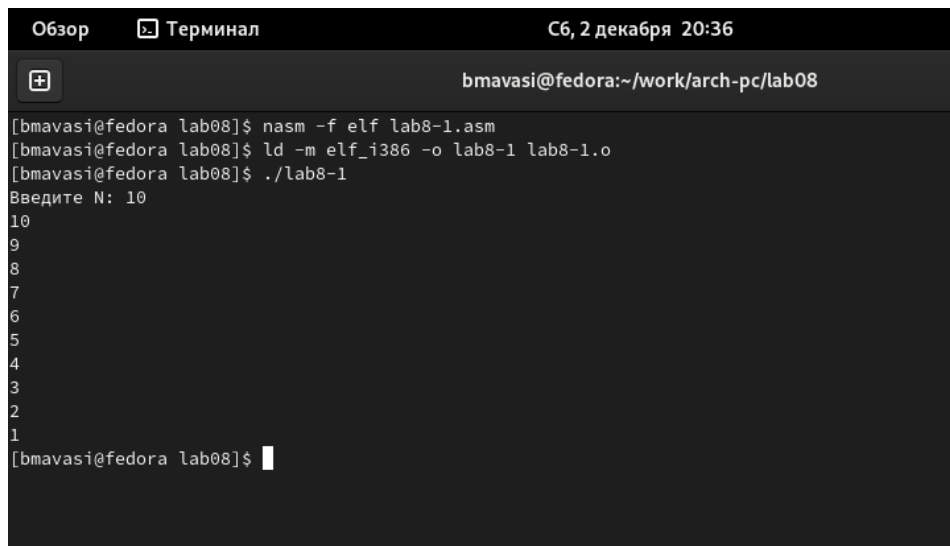
Открываю созданный файл `lab8-1.asm`, вставляю в него программу с использованием цикла для вывода чисел(рис. [2.2]).

```
Обзор Терминал Сб, 2 декабря 20:32
bmavasi@fedora:~/work/arch-pc/lab08 — nano lab8-1.asm
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
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1`,N-1 и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.2: Редактирование файла

Шаг 3

Создаю исполняемый файл программы и запускаю его (рис. [2.3]).



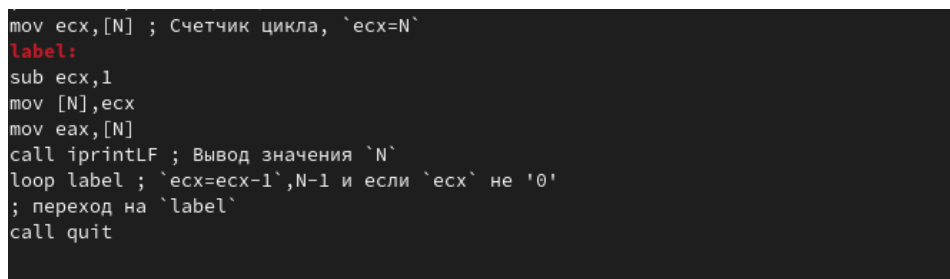
```
Обзор Терминал С6, 2 декабря 20:36
bmavasi@fedora:~/work/arch-pc/lab08

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

Рис. 2.3: Запуск исполняемого файла

Шаг 4

с помощью инструкции `sub` уменьшаю изначальный индекс на 1 единицу. (рис. [2.4]).



```
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1`, N-1 и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.4: Уменьшение индекса

Шаг 5

Создаю новый исполняемый файл программы и запускаю его (рис. [2.5]). Получаем результат отличный от ожидаемого

```
[bmavasi@fedora lab08]$ nasm -f elf lab8-1.asm
[bmavasi@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[bmavasi@fedora lab08]$ ./lab8-1
Введите N: 10
9
7
5
3
1
[bmavasi@fedora lab08]$
```

Рис. 2.5: Запуск исполняемого файла

Шаг 6

Изменяю текст программы так, чтобы получить нужный результат, используя стеки для запоминания данных. (рис. [2.6]).

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

Рис. 2.6: Редактирование программы

Шаг 7

Создаю исполняемый файл и проверяю работу программы (рис. [2.7]).


```

[bmavasi@fedora lab08]$ nasm -f elf lab8-1.asm
[bmavasi@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[bmavasi@fedora lab08]$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
[bmavasi@fedora lab08]$

```

Рис. 2.7: Создание исполняемого файла

- Программа отработало верно.

Шаг 8

Создаю новый файл lab8-2.asm для новой программы. (рис. [2.8]).

```

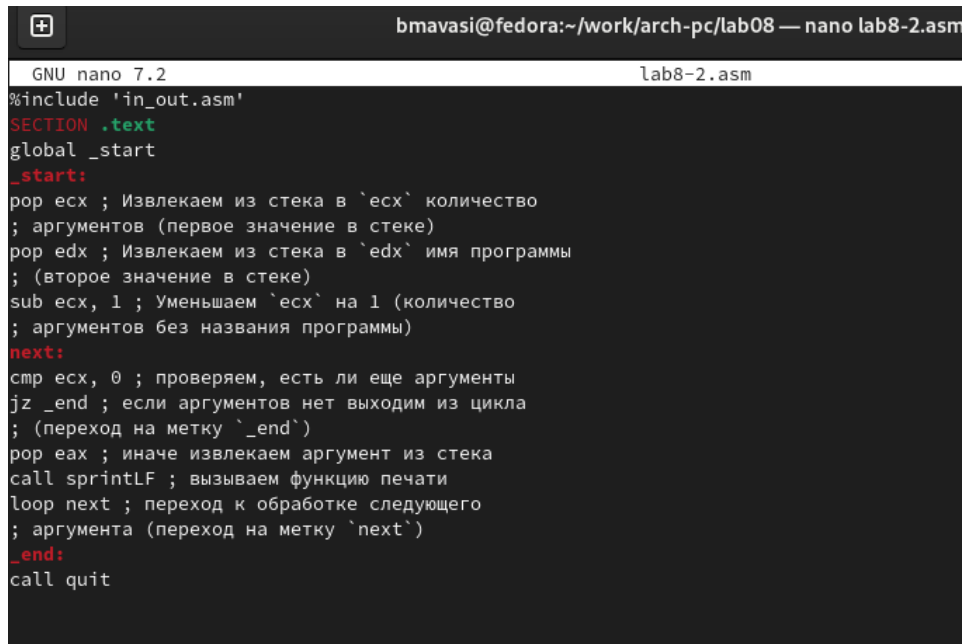
bmavasi@fedora:~/work/arch-pc/lab08
[bmavasi@fedora lab08]$ touch lab8-2.asm
[bmavasi@fedora lab08]$ nano lab8-2.asm
[bmavasi@fedora lab08]$

```

Рис. 2.8: Создание файла

Шаг 9

Вставляю программу, которая выводит все введенные пользователем аргументы (рис.[2.9]).

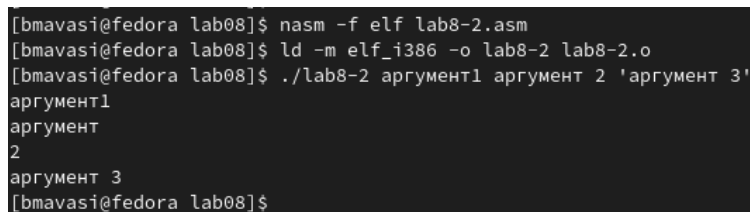


```
GNU nano 7.2 lab8-2.asm
%include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
; аргументов 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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 2.9: Вставляю текст в файл

Шаг 10

Создаю и запускаю новый исполняемый файл, проверяю работу программы (рис. [2.10]).



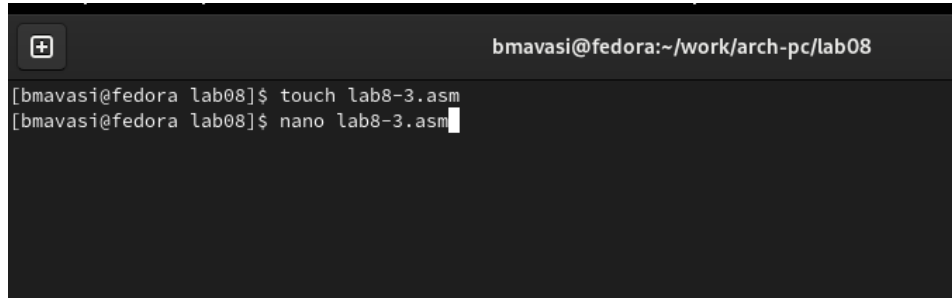
```
[bmavasi@fedora lab08]$ nasm -f elf lab8-2.asm
[bmavasi@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[bmavasi@fedora lab08]$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[bmavasi@fedora lab08]$
```

Рис. 2.10: Запуск исполняемого файла

- Программой было обработано 4 аргумента.
- Программа считает аргументами все символы до пробела, или значения, которые взяты в кавычки.

Шаг 11

Создаю новый файл lab8-3.asm (рис. [2.11]).

A terminal window with a dark background. The title bar shows a window icon and the text 'bmavasi@fedora:~/work/arch-pc/lab08'. The terminal contains two lines of text: the first line is '[bmavasi@fedora lab08]\$ touch lab8-3.asm' and the second line is '[bmavasi@fedora lab08]\$ nano lab8-3.asm' with a white cursor at the end of the command.

```
[bmavasi@fedora lab08]$ touch lab8-3.asm  
[bmavasi@fedora lab08]$ nano lab8-3.asm
```

Рис. 2.11: Создание файла

Шаг 12

Открываю файл и ввожу программу, которая складывает все числа введенные пользователем (рис. [2.12]).

```
bmavasi@fedora:~/work/arch-pc/lab08 — nano lab8-3.asm
GNU nano 7.2 lab8-3.asm
#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 произведений
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 промежуточному произведению текущее
; 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 iprintf ; печать результата
call quit ; завершение программы
```

Рис. 2.12: вставляю программу

Шаг 13

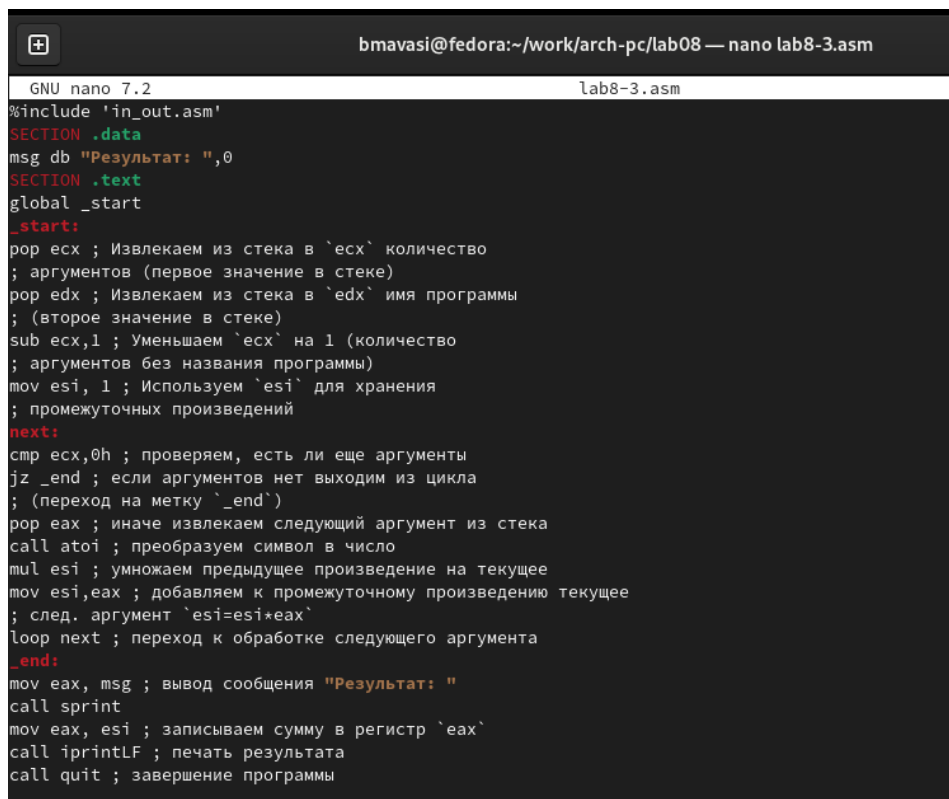
Запускаю исполняемый файл и проверяю работу программы (рис. [2.13]).

```
[bmavasi@fedora lab08]$ nasm -f elf lab8-3.asm
[bmavasi@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[bmavasi@fedora lab08]$ ./lab8-3 5 10 25 30
Результат: 70
[bmavasi@fedora lab08]$
```

Рис. 2.13: Запуск программы

Шаг 14

Изменяю текст программы так, чтобы она выводила произведение всех чисел, введенные пользователем. (рис. [2.14]).

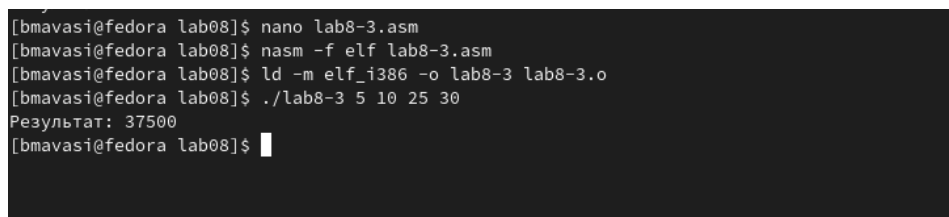


```
GNU nano 7.2 lab8-3.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем `esi` для хранения
; промежуточных произведений
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 ; умножаем предыдущее произведение на текущее
mov esi,eax ; добавляем к промежуточному произведению текущее
; 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
```

Рис. 2.14: Редактирование файла

Шаг 15

Запускаю исполняемый файл и проверяю работу программы (рис. [2.15]).



```
[bmavasi@fedora lab08]$ nano lab8-3.asm
[bmavasi@fedora lab08]$ nasm -f elf lab8-3.asm
[bmavasi@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[bmavasi@fedora lab08]$ ./lab8-3 5 10 25 30
Результат: 37500
[bmavasi@fedora lab08]$
```

Рис. 2.15: Запуск программы

Программа отработала верно!

3 Самостоятельная работа

Шаг 1

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

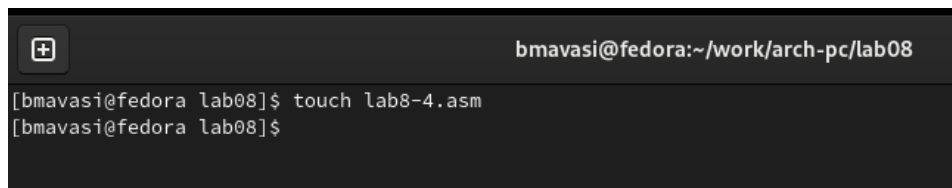
A terminal window with a dark background. The title bar shows a window icon and the text 'bmavasi@fedora:~/work/arch-pc/lab08'. The terminal content shows two lines of text: '[bmavasi@fedora lab08]\$ touch lab8-4.asm' followed by a new prompt line '[bmavasi@fedora lab08]\$'.

Рис. 3.1: Создание файла

Шаг 2

Ввожу в созданный файл текст программы, у, которая находит сумму значений функции (2 Вариант) $f(x)=3x-1$ для всех аргументов x , введенные пользователем.(рис. [3.2]).

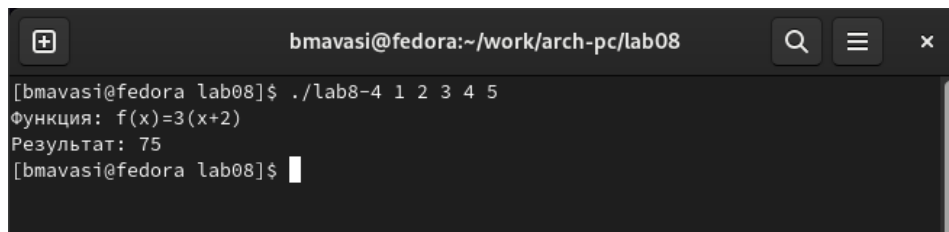
```
GNU nano 7.2 lab8-4
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg1 db "Функция: f(x)=3(x+2)"
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение 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 eax,2 ;
mov ebx,3 ;ebx=3
mul ebx; eax=eax*ebx
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax,msg1 ;
call printf ;
mov eax, msg ; вывод сообщения "Результат: "
call printf
mov eax, esi ; записываем сумму в регистр `eax`
```

[Прочитано 3]

Рис. 3.2: Редактирование файла

Шаг 3

Создаю исполняемый файл и запускаю его, при $x = 5, 3, 6$ (рис. [3.3]).



```
[bmavasi@fedora lab08]$ ./lab8-4 1 2 3 4 5
Функция: f(x)=3(x+2)
Результат: 75
[bmavasi@fedora lab08]$
```

Рис. 3.3: Запуск исполняемого файла

Текст программы

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0
msg1 db "Функция: f(x)=3(x+2)"

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 eax,2 ; eax+2
mov ebx,3 ;ebx=3
mul ebx; eax=eax*ebx
add esi,ebx ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax,msg1 ;
call sprintf ;
mov eax, msg ; вывод сообщения "Результат: "
call sprintf
mov eax, esi ; записываем сумму в регистр `eax`
call printf ; печать результата
call quit ; завершение программы
```

4 Вывод

В ходе выполнения работы были получены навыки по организации циклов и по работе со стеком на языке NASM.