

# **Отчёт по лабораторной работе №8**

**дисциплина: Архитектура компьютера**

Кудинец Максим Антонович

# 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>
4.1	4.1 Реализация циклов в NASM . . . . .	8
4.2	4.2 Обработка аргументов командной строки . . . . .	10
<b>5</b>	<b>5 Задания для самостоятельной работы</b>	<b>14</b>
<b>6</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

# Список иллюстраций

4.1	Переход в каталог и создание файла . . . . .	8
-----	--	---

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

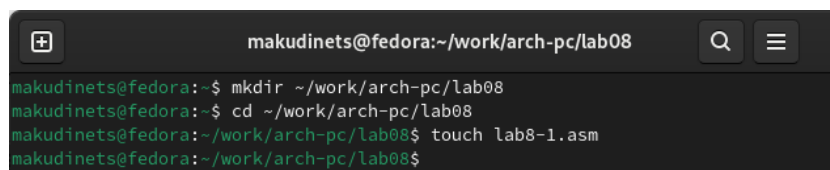
1. Напишите программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x_i$  передаются как аргументы. Вид функции  $f(x)$  выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах  $x = x_1, x_2, \dots, x_n$ .

### **3 Теоретическое введение**

## 4 Выполнение лабораторной работы

### 4.1 4.1 Реализация циклов в NASM

1. Создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm.

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

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

Рис. 4.1: Переход в каталог и создание файла

2. Ввожу в файл lab8-1.asm текст программы из листинга 8.1. Запускаю исполняемый файл.



```

%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` и если `ecx` не '0'
; переход на `label`
call quit

```

```

makudinets@fedora:~/work/arch-pc/lab08$ nasm -f elf32 lab8-1.asm
makudinets@fedora:~/work/arch-pc/lab08$ ld -m elf32 -o lab8-1
makudinets@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
makudinets@fedora:~/work/arch-pc/lab08$

```

3. Изменим текст программы, добавив изменение значение регистра ecx в цикле. Запустим исправленную программу. Число проходов цикла не соответствует значению, введенному с клавиатуры.

```

; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
    sub ecx,1 ; `ecx=ecx-1`
    mov [N],ecx
    mov eax,[N]
    call iprintLF
    loop label
; переход на `label`
call quit

```

```

makudinets@fedora:~/work/a
makudinets@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
makudinets@fedora:~/work/arch-pc/lab08$

```

4. Внесем изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. Запустим программу и проверим ее работу. Теперь число проходов цикла соответствует числу, введенному с клавиатуры.

```

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

```

```

makudinets@fedora:~/work/arch-pc/lab08$ nasm -f elf
makudinets@fedora:~/work/arch-pc/lab08$ ld -m elf
makudinets@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
makudinets@fedora:~/work/arch-pc/lab08$

```

## 4.2 4.2 Обработка аргументов командной строки

5. Создаем файл lab8-2.asm. Вводим в него программу из листинга 8.2. Программа обработала 4 аргумента.



```

%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 sprint
    mov eax, esi       ; записываем сумму в регистр `eax`
    call iprintLF     ; печать результата
    call quit         ; завершение программы

```

```

makudinets@fedora:~/work/arch-pc/lab08$ touch lab
makudinets@fedora:~/work/arch-pc/lab08$ nasm -f elf32 lab.asm
makudinets@fedora:~/work/arch-pc/lab08$ ld -m elf32 -o lab8-3
makudinets@fedora:~/work/arch-pc/lab08$ ./lab8-3
Результат: 76
makudinets@fedora:~/work/arch-pc/lab08$

```

7. Изменяю текст программы для вычисления произведения аргументов командной строки.

```

%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
    mov ebx, esi
    mul ebx
    mov esi, eax
    loop next
_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit

```

```

makudinets@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
makudinets@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
makudinets@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 3 6
Результат: 90
makudinets@fedora:~/work/arch-pc/lab08$

```

## 5 5 Задания для самостоятельной работы

1. Напишем программу, которая находит сумму значений функции  $f(x)$  для  $x=x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Мой вариант - 10. Создадим исполняемый файл и проверим его работу на нескольких наборах  $x=x_1, x_2, \dots, x_n$ . Программа работает корректно.

```

%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 5 * (2 + x)", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintfLF

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

add esi, eax

loop next

_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintLF
call quit

```

```

makudinets@fedora:~/work/arch-p
makudinets@fedora:~/work/arch-p/lab08$ touch lab8-4
makudinets@fedora:~/work/arch-p/lab08$ nasm -f elf
makudinets@fedora:~/work/arch-p/lab08$ ld -m elf_i386
makudinets@fedora:~/work/arch-p/lab08$ ./lab8-4
Функция: f(x) = 5 * (2 + x)
Результат: 0
makudinets@fedora:~/work/arch-p/lab08$ ./lab8-4 3
Функция: f(x) = 5 * (2 + x)
Результат: 25
makudinets@fedora:~/work/arch-p/lab08$ ./lab8-4 3 2
Функция: f(x) = 5 * (2 + x)
Результат: 75
makudinets@fedora:~/work/arch-p/lab08$

```

## 6 Выводы

В результате выполнения лабораторной работы я приобрёл навыки написания программ с использованием циклов и обработкой аргументов командной строки в NASM.



## **Список литературы**