

# **Отчет по лабораторной работе №8**

**дисциплина: Архитектура компьютера**

Бондарь Татьяна Владимировна

# 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	Реализация циклов в NASM . . . . .	8
4.2	Обработка аргументов командной строки . . . . .	11
<b>5</b>	<b>Задания для самостоятельной работы</b>	<b>14</b>
<b>6</b>	<b>Выводы</b>	<b>16</b>

# Список иллюстраций

4.1	Переход в каталог и создание файла . . . . .	8
4.2	Программа вывода значений регистра есх . . . . .	9
4.3	Исполнение программы из листинга 8.1 . . . . .	9
4.4	Исправленный текст программы lab8-1.asm . . . . .	10
4.5	Исполнение программы lab8-1 . . . . .	10
4.6	Исправленный текст программы lab8-1.asm . . . . .	10
4.7	Исполнение программы lab8-1.asm . . . . .	11
4.8	Текст программы из листинга 8.2 . . . . .	11
4.9	Исполнение программы . . . . .	11
4.10	Текст программы из листинга 8.3 . . . . .	12
4.11	Исполнение программы . . . . .	12
4.12	Измененный текст программы из листинга 8.3 . . . . .	12
4.13	Исполнение программы . . . . .	13
5.1	Текст программы lab8-4.asm . . . . .	14
5.2	Запуск программы . . . . .	15

## **Список таблиц**

# **1 *Цель работы***

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

## 2 Задание

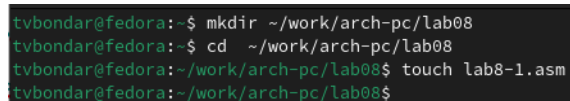
1. . Напишите программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x_i$  передаются как аргументы. Вид функции  $f(x)$  выбрать из таблицы 8.1 вариантов заданий в соответствии 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 *Реализация циклов в NASM*

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



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



```

; 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:
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

```

Рис. 4.2: Программа вывода значений регистра ecx

```

tvbondar@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
tvbondar@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
tvbondar@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
tvbondar@fedora:~/work/arch-pc/lab08$

```

Рис. 4.3: Исполнение программы из листинга 8.1

- Изменим текст программы, добавив изменение значение регистра ecx в цикле. Запустим исправленную программу. Регистр ecx в цикле принимает значения КАКИЕ БЛЯТИЬ. Число проходов цикла не соответствует значению,

введенному с клавиатуры.

```
mov ecx, [N]      ; СЧЕТЧИК ЦИКЛА, ecx=N
label:
    sub ecx, 1      ; `ecx=ecx-1`
    mov [N], ecx
    mov eax, [N]
    call iprintfLF
loop label
    ; ПЕРЕХОД НА `label`
call quit
```

Рис. 4.4: Исправленный текст программы lab8-1.asm

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

Рис. 4.5: Исполнение программы lab8-1

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 iprintfLF
    pop ecx         ; извлечение значения ecx из стека
loop label
    ; переход на `label`
call quit
```

Рис. 4.6: Исправленный текст программы lab8-1.asm

```

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

```

Рис. 4.7: Исполнение программы lab8-1.asm

## 4.2 Обработка аргументов командной строки

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

```

; Обработка аргументов командной строки
#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. 4.8: Текст программы из листинга 8.2

```

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

```

Рис. 4.9: Исполнение программы

6. Создадим файл lab8-3.asm и введем в него текст программы из листинга 8.3.

```

;Листинг 8.3. Программа вычисления суммы аргументов командной строки
#include "jn_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,0b      ; 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
    call quit        ; завершение программы

```

Рис. 4.10: Текст программы из листинга 8.3

```

tvbondar@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
tvbondar@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
tvbondar@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
tvbondar@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 4 1 45
Результат: 62
tvbondar@fedora:~/work/arch-pc/lab08$

```

Рис. 4.11: Исполнение программы

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

```

;Листинг 8.3. Программа вычисления произведения аргументов командной строки
#include "jn_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, начинаем с 1 чтобы не умножать на 0

next:
    cmp ecx,0b      ; 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
    mov ebx, esi     ; сохраняем esi в ebx
    mul ebx          ; умножаем промежуточное произведение на след. аргумент 'esi=esi*eax'
    mov esi, eax     ; возвращаем результат в esi
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
    call quit        ; завершение программы

```

Рис. 4.12: Измененный текст программы из листинга 8.3

```
tvbondar@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
tvbondar@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
tvbondar@fedora:~/work/arch-pc/lab08$ ./lab8-3 2 3
Результат: 6
tvbondar@fedora:~/work/arch-pc/lab08$
```

Рис. 4.13: Исполнение программы

## 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 - 12. Создадим исполняемый файл и проверим его работу на нескольких наборах  $x = x_1, x_2, \dots, x_n$ . Программа работает корректно.

```
;ЛИСТИНГ 8.3. Программа вычисления суммы значений функции  $f(x)=15x-9$ 
#include <io_out.asm>
SECTION .data
msg1 db "Функция:  $f(x)=15x-9$  ",0
msg2 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
    mov ebx, 15
    mul ebx          ; умножаем переменную на 15 'eax=15*eax'
    sub eax, 9       ; уменьшаем значение на 9
    add esi, eax      ; возвращаем результат в esi
    loop next        ; переход к обработке следующего аргумента
_end:
    mov eax, msg1     ; вывод первого сообщения
    call sprintf
    mov eax, msg2     ; вывод второго сообщения
    call sprintf
    mov eax, esi      ; записываем сумму в регистр 'eax'
    call printf
    call quit        ; завершение программы
```

Рис. 5.1: Текст программы lab8-4.asm

```
tvbondar@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
tvbondar@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
tvbondar@fedora:~/work/arch-pc/lab08$ ./lab8-4 2 3 4
Функция:  $f(x)=15x-9$  Результат: 108
tvbondar@fedora:~/work/arch-pc/lab08$ ./lab8-4 15 12 24
Функция:  $f(x)=15x-9$  Результат: 738
tvbondar@fedora:~/work/arch-pc/lab08$ ./lab8-4 100 1000 20000
Функция:  $f(x)=15x-9$  Результат: 316473
tvbondar@fedora:~/work/arch-pc/lab08$
```

Рис. 5.2: Запуск программы

## **6 Выводы**

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