

# **Отчёт по лабораторной работе №8**

**дисциплина: Архитектура вычислительных систем**

Новичков максим Алексеевич

# 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
2.2	Обработка аргументов командной строки . . . . .	9
2.3	Задание для самостоятельной работы . . . . .	13
<b>3</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

2.1	Создаем каталог для работы . . . . .	6
2.2	Текст программы в соответствии с листингом 8.1 . . . . .	6
2.3	Вывод программы из листинга 8.1 . . . . .	7
2.4	Изменённый текст программы в соответствии с листингом 8.1 . .	8
2.5	Вывод изменённой программы из листинга 8.1 . . . . .	8
2.6	Текст программы lab8-2 . . . . .	9
2.7	Вывод программы lab8-2 . . . . .	10
2.8	Текст программы из листинга 8.3 . . . . .	11
2.9	Проверяем работу программы из листинга 8.3 . . . . .	11
2.10	Изменённый текст программы lab8-3 . . . . .	12
2.11	Вывод изменённой программы lab8-3 . . . . .	12
2.12	Текст программы из задания для самостоятельного выполнения (lab8-4) . . . . .	14
2.13	Вывод программы из задания для самостоятельного выполнения	14

## Список таблиц

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

## 2 Выполнение лабораторной работы

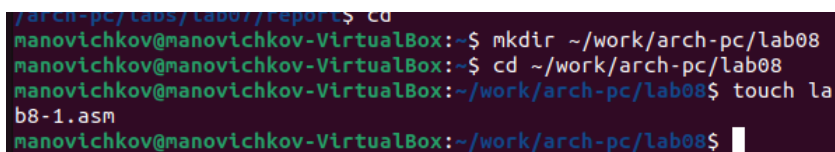
### 2.1 Реализация циклов в NASM

Создаём каталог для программ лабораторной работы № 8, переходим в него и создаём файл *lab8-1.asm*. С помощью команд:

```
mkdir ~/work/arch-pc/lab08
```

```
cd ~/work/arch-pc/lab08
```

```
touch lab8-1.asm
```

A screenshot of a terminal window with a dark background and light-colored text. The terminal shows a series of commands being executed in a shell. The first command is 'mkdir ~/work/arch-pc/lab08', followed by 'cd ~/work/arch-pc/lab08', and then 'touch lab8-1.asm'. The prompt for each command is 'manovichkov@manovichkov-VirtualBox:~\$'. The final prompt is 'manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08\$' with a cursor at the end.

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

Рис. 2.1: Создаем каталог для работы

При реализации циклов в *NASM* с использованием инструкции *loop* необходимо помнить о том, что эта инструкция использует регистр *ecx* в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра *ecx*. Внимательно изучим текст программы (*Листинг 8.1*).

Текст программы в соответствии с листингом 8.1

Рис. 2.2: Текст программы в соответствии с листингом 8.1

Для того, чтобы программа транслировалась без ошибок перенесем файл *in\_out.asm* в *~/work/arch-pc/lab08*.

Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим:

```
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ nasm -f
elf lab8-1.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ld -m el
f_i386 -o lab8-1 lab8-1.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.3: Вывод программы из листинга 8.1

Данный пример показывает, что использование регистра *ecx* в теле цикла *loop* может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра *ecx* в цикле:

```

1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 sub ecx,1 ; `ecx=ecx-1`
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF
30 loop label ; `ecx=ecx-1` и если `ecx` не '0'
31 ; переход на `label`
32 call quit

```

Рис. 2.4: Изменённый текст программы в соответствии с листингом 8.1

Транслируем текст, создаём объектный файл, компилируем его и запускаем программу *lab8-1*:

```

manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$

```

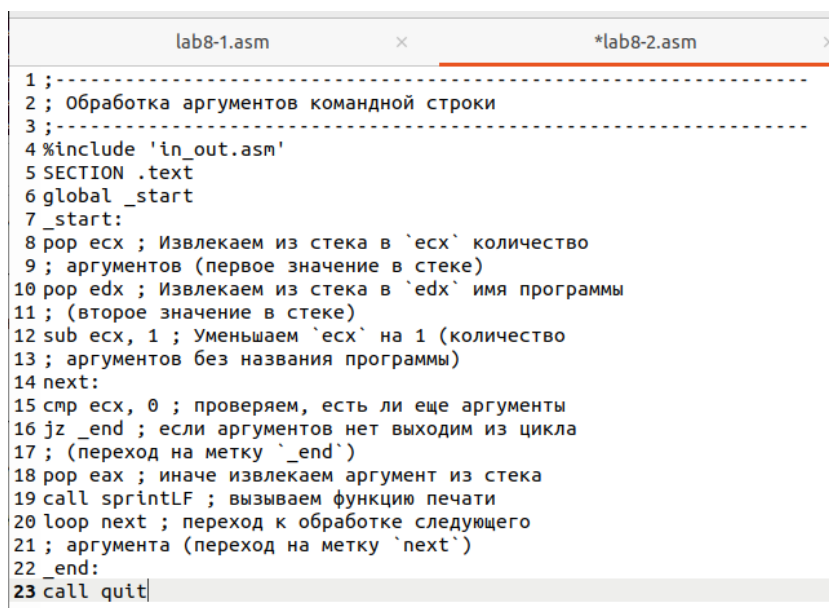
Рис. 2.5: Вывод изменённой программы из листинга 8.1

Число проходов цикла соответствует значению  $N$  введенному с клавиатуры, регистр *ecx* принимает значения все нечётные значения от 0 до  $N$ .



## 2.2 Обработка аргументов командной строки

Для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки. Внимательно изучим текст программы (Листинг 8.2).



```
lab8-1.asm  x  *lab8-2.asm  x
1 ;-----
2 ; Обработка аргументов командной строки
3 ;-----
4 %include 'in_out.asm'
5 SECTION .text
6 global _start
7 _start:
8 pop ecx ; Извлекаем из стека в `ecx` количество
9 ; аргументов (первое значение в стеке)
10 pop edx ; Извлекаем из стека в `edx` имя программы
11 ; (второе значение в стеке)
12 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
13 ; аргументов без названия программы)
14 next:
15 cmp ecx, 0 ; проверяем, есть ли еще аргументы
16 jz _end ; если аргументов нет выходим из цикла
17 ; (переход на метку `_end`)
18 pop eax ; иначе извлекаем аргумент из стека
19 call sprintf ; вызываем функцию печати
20 loop next ; переход к обработке следующего
21 ; аргумента (переход на метку `next`)
22 _end:
23 call quit
```

Рис. 2.6: Текст программы lab8-2

Транслируем текст, создаём объектный файл, компилируем его и запускаем программу *lab8-2*:

```
nasm -f elf lab8-2.asm
```

```
ld -m elf_i386 -o lab8-2 lab8-2.o
```

```
./lab8-2 1 2 'Hi'
```

```

manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ nasm -f
elf lab8-2.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ld -m el
f_i386 -o lab8-2 lab8-2.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2
2 4 6
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2
2 4 6
bash: ./lab8-22: Нет такого файла или каталога
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2
2 4 6
2
4
6
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 2.7: Вывод программы lab8-2

Программа обрабатывает 3 аргумента.

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл *lab8-3.asm* в каталоге *~/work/arch-pc/lab08* и введём в него текст программы из листинга 8.3.

```
lab8-1.asm  x      lab8-2.asm  x      *lab8-3.asm
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент `esi=esi+eax`
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintf ; печать результата
29 call quit ; завершение программы
```

Текст ▾    Ширина табуляции: 8 ▾    Стр 29, Стлб 33 ▾

Рис. 2.8: Текст программы из листинга 8.3

Создадим исполняемый файл и запустим его, указав аргументы.

```
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ nasm -f
elf lab8-3.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ld -m el
f_i386 -o lab8-3 lab8-3.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3
Результат: 0
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3
23 65 911
Результат: 999
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.9: Проверяем работу программы из листинга 8.3

Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в ecx количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в edx имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем ecx на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 1 ; Используем esi для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку _end)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 mul esi ; добавляем к промежуточной сумме
22 mov esi,eax
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр eax
28 call iprintLF ; печать результата
29 call quit ; завершение программы
```

Matlab ▾ Ширина табуляции: 8 ▾ Стр 29, Стлб 33 ▾

Рис. 2.10: Изменённый текст программы lab8-3

Проверим работу программы:

```
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ nasm -f
elf lab8-3.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ld -m el
f_i386 -o lab8-3 lab8-3.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3
12 4 67
Результат: 3216
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.11: Вывод изменённой программы lab8-3

А теперь на вывод программы с использованием *iprint*.

## 2.3 Задание для самостоятельной работы

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 (*в моём случае вариант 9*), полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах  $x = x_1, x_2, \dots, x_n$ .

Напишем текст программы для  $f(x) = 10x - 4$  (вариант 9):

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7
8 pop ecx
9
10 pop edx
11
12 sub ecx,1
13
14 mov esi,0
15
16 next:
17 cmp ecx,0h
18 jz _end
19
20 pop eax
21 call atoi
22 mov ebx, 10
23 mul ebx
24 sub eax, 4
25 add eax, esi
26 mov esi, eax
27
28 loop next
29 _end:
30 mov eax, msg
31 call sprint
32 mov eax, esi
33 call iprintLF
34 call quit

```

Сохранен... Matlab ▾ Ширина табуляции: 8 ▾ Стр 14, Стлб 8 ▾ ВСТ

Рис. 2.12: Текст программы из задания для самостоятельного выполнения (*lab8-4*)

Убедимся в правильности написанной программы.

```

manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 3 2 4
Результат: 78
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 2.13: Вывод программы из задания для самостоятельного выполнения

## **3 Выводы**

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