

# **РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук Кафедра  
прикладной информатики и теории вероятностей**

Архипов Олег Константинович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
2.1	Реализация циклов в NASM . . . . .	5
2.2	Обработка аргументов командной строки . . . . .	9
<b>3</b>	<b>Задание для самостоятельной работы</b>	<b>14</b>
<b>4</b>	<b>Выводы</b>	<b>17</b>

## Список иллюстраций

2.1	Папка и файл ЛР . . . . .	5
2.2	Программа вывода значений регистра 'есх' . . . . .	6
2.3	Работа программы lab8-1 . . . . .	7
2.4	Изменение 1 в файле lab8-1.asm . . . . .	7
2.5	Работа измененной первый раз программы lab8-1 . . . . .	8
2.6	Изменение 2 в файле lab8-1.asm . . . . .	8
2.7	Работа измененной второй раз программы lab8-1 . . . . .	9
2.8	Файл lab8-2.asm . . . . .	9
2.9	Текст программы lab8-2 . . . . .	10
2.10	Работа программы lab8-2 . . . . .	10
2.11	Файл lab8-3.asm . . . . .	11
2.12	Код lab8-3 . . . . .	11
2.13	Работа программы lab8-3 . . . . .	12
2.14	Измененный код lab8-3 . . . . .	13
2.15	Работа измененной программы lab8-3 . . . . .	13
3.1	Файл sol8.asm . . . . .	14
3.2	Результаты работы sol8 . . . . .	16

# 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

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm (рис. 2.1).

```
[okarkhipov@fedora ~]$ mkdir ~/work/arch-pc/lab08  
[okarkhipov@fedora ~]$ cd ~/work/arch-pc/lab08  
[okarkhipov@fedora lab08]$ touch lab8-1.asm  
[okarkhipov@fedora lab08]$
```

Рис. 2.1: Папка и файл ЛР

Изучаю программу, которая выводит значение регистра есх, являющегося счетчиком для инструкции loop (рис. 2.2).

```

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

```

Рис. 2.2: Программа вывода значений регистра 'ecx'

Создаю исполняемый файл и проверяю его работу для значения 10, однако программа работает не так, как задумывалось: она должна выводить значения от 9 до 0 (рис. 2.3).

```

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

```

Рис. 2.3: Работа программы lab8-1

Изменяю текст программы, добавляя изменение значения регистра `ecx` в цикле (рис. 2.4).

```

label:
sub ecx,1 ; `ecx=ecx-1`
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. 2.4: Изменение 1 в файле lab8-1.asm

Еще раз проверяю работу программы, снова получаю неверный результат, т.к. теперь программа за 1 проход цикла дважды уменьшает значение `ecx` на 1. Т.е. получаю первое значение 9, а далее каждый раз вычитается 2: 7, 5, 3, 1 (рис. 2.5).

```

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

```

Рис. 2.5: Работа измененной первый раз программы lab8-1

Для сохранения корректности работы программы с использованием регистра есх в цикле нужен стек: ввожу команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис. 2.6).

```

label:
push есх ; добавление значения есх в стек
sub есх,1 ; `есх=есх-1`
mov [N],есх
mov еах,[N]
call iprintLF ; Вывод значения `N`
pop есх ; извлечение значения есх из стека
loop label ; `есх=есх-1` и если `есх` не '0'
; переход на `label`
call quit

```

Рис. 2.6: Изменение 2 в файле lab8-1.asm

Проверяю работу программы, на этот раз результат корректен (рис. 2.7).



```

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

```

Рис. 2.7: Работа измененной второй раз программы lab8-1

## 2.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm (рис. 2.8).

```

[okarkhipov@fedora lab08]$ touch lab8-2.asm
[okarkhipov@fedora lab08]$

```

Рис. 2.8: Файл lab8-2.asm

Ввожу текст программы обработки аргументов командной строки (рис. 2.9).

```

#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: Текст программы lab8-2

Создаю исполняемый файл и запускаю его с аргументами 9, 5, '3', т.е. программа обработала 3 аргумента, а также еще два дополнительных: имя программы и количество элементов (рис. 2.10).

```

[okarkhipov@fedora lab08]$ nasm -f elf lab8-2.asm
[okarkhipov@fedora lab08]$ ld -m elf_i386 lab8-2.o -o lab8-2
[okarkhipov@fedora lab08]$ ./lab8-2
[okarkhipov@fedora lab08]$ ./lab8-2 9 5 '3'
9
5
3
[okarkhipov@fedora lab08]$ 

```

Рис. 2.10: Работа программы lab8-2

Создаю файл lab8-3.asm (рис. 2.11).

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

Рис. 2.11: Файл lab8-3.asm

Пишу код программы которая выводит сумму чисел, которые передаются в программу как аргументы (рис. 2.12).

```
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 iprintLF ; печать результата  
29 call quit ; завершение программы
```

Рис. 2.12: Код lab8-3

Далее создаю исполняемый файл с названием main и аргументами '12 13 7 10 5' и запускаю его (рис. 2.13).

```
[okarkhipov@fedora lab08]$ nasm -f elf lab8-3.asm
[okarkhipov@fedora lab08]$ ld -m elf_i386 lab8-3.o -o main
[okarkhipov@fedora lab08]$ ./main 12 13 7 10 5
Результат: 47
[okarkhipov@fedora lab08]$
```

Рис. 2.13: Работа программы lab8-3

Меняю текст программы для вычисления произведения аргументов командной строки (рис. 2.14). Для этого вместо

```
mov esi, 0
```

пишу

```
mov esi, 1
```

т.к. нейтральный элемент по умножению - единица, а также вместо

```
add esi, eax
```

пишу

```
mul esi
```

```
mov esi, eax
```

```

11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 1 ; Используем `esi` для хранения
14 ; промежуточных произведений
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 ; умножаем на промежуточное произведение
22 mov esi, eax ; копируем промежуточный результат умножения
23 ; в регистр esi
24 ; след. аргумент `esi=esi+eax`
25 loop next ; переход к обработке следующего аргумента
26 _end:
27 mov eax, msg ; вывод сообщения "Результат: "
28 call sprint
29 mov eax, esi ; записываем произведение в регистр `eax`
30 call iprintLF ; печать результата
31 call quit ; завершение программы

```

Рис. 2.14: Измененный код lab8-3

После исполнения обновленной программы получаю 54600 (рис. 2.15).

```

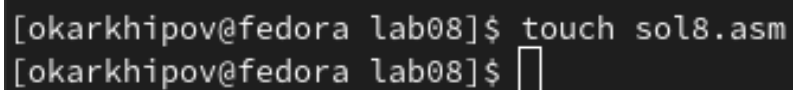
[okarkhipov@fedora lab08]$ nasm -f elf lab8-3.asm
[okarkhipov@fedora lab08]$ ld -m elf_i386 lab8-3.o -o main
[okarkhipov@fedora lab08]$ ./main 12 13 7 10 5
Результат: 54600
[okarkhipov@fedora lab08]$ 

```

Рис. 2.15: Работа измененной программы lab8-3

### 3 Задание для самостоятельной работы

Создаю файл sol8.asm (рис. 3.1).



```
[okarkhipov@fedora lab08]$ touch sol8.asm
[okarkhipov@fedora lab08]$
```

Рис. 3.1: Файл sol8.asm

Беру тот же вариант, что был в двух предыдущих ЛР - 4. Код программы приведен в листинге ниже.

#### Листинг программы для вычисления суммы значений функции

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0
f db "Функция: f(x)=2(x-1)",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
sub eax, 1 ; eax=eax-1
mov edi, 2 ; edi=2
mul edi    ; eax=eax*edi
add esi, eax ; прибавляем к промежуточной сумме
loop next  ; переход к обработке следующего аргумента
_end:
mov eax, f ; вывод сообщения "Функция: f(x)=2(x-1)"
call sprintf
mov eax,msg ; вывод сообщения "Результат: "
call sprintf
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit   ; завершение программы

```

Проверяю работу программы с разными значениями x (рис. 3.2).

```
[okarkhipov@fedora lab08]$ nasm -f elf sol8.asm
[okarkhipov@fedora lab08]$ ld -m elf_i386 sol8.o -o main1
[okarkhipov@fedora lab08]$ ./main1 14 2 73 9
Функция:  $f(x)=2(x-1)$ 
Результат: 188
[okarkhipov@fedora lab08]$ ./main1 1 2 3
Функция:  $f(x)=2(x-1)$ 
Результат: 6
[okarkhipov@fedora lab08]$
```

Рис. 3.2: Результаты работы sol8



## 4 Выводы

Была освоена работа со стеком, циклами и обработка аргументов командной строки, а также методы написания программ при помощи данных инструментов.