

# **Лабораторная работа №8**

**Программирование цикла. Обработка аргументов командной строки**

Налобин Михаил Дмитриевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Ход работы</b>	<b>6</b>
<b>3</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

2.1	Создание каталога lab08 . . . . .	6
2.2	Код программы lab8-1.asm . . . . .	7
2.3	Запуск программы lab8-1 . . . . .	8
2.4	Измененная часть кода программы lab8-1.asm . . . . .	8
2.5	Запуск измененной программы lab8-1 . . . . .	9
2.6	Повторно измененная часть кода программы lab8-1.asm . . . . .	9
2.7	Запуск повторно измененной программы lab8-1 . . . . .	10
2.8	Создание файла lab8-2.asm . . . . .	10
2.9	Код программы lab8-2.asm . . . . .	11
2.10	Запуск программы lab8-2 . . . . .	11
2.11	Создание файла lab8-3.lst . . . . .	12
2.12	Код программы lab8-3.asm . . . . .	13
2.13	Запуск программы lab8-3 . . . . .	14
2.14	Код измененной программы lab8-3.asm . . . . .	15
2.15	Запуск измененной программы lab8-3 . . . . .	16
2.16	Создание файла sr.asm . . . . .	16
2.17	Код программы sr.asm . . . . .	17
2.18	Запуск программы sr . . . . .	18

## **Список таблиц**

# 1 Цель работы

Освоить навык работы с циклами и обработкой аргументов командой строки в программах на языке ассемблера NASM.

## 2 Ход работы

Создали каталог lab08 для файлов лабораторной работы №8 и в нем файл lab8-1.asm (рис. 2.1).

```
mdnalogin@dk3n62 ~/work/arch-pc $ mkdir lab08
mdnalogin@dk3n62 ~/work/arch-pc $ cd lab08
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 2.1: Создание каталога lab08

Переписали в него текст из Листинга 8.1. (рис. 2.2).

```

#include 'in_out.asm'

SECTION .data
    msg1: DB 'Введите N: ', 0h

SECTION .bss
    N:     resb 10

SECTION .text
GLOBAL _start
_start:

    mov eax, msg1
    call sprint

    mov ecx, N
    mov edx, 10
    call sread

    mov eax, N
    call atoi
    mov [N], eax

    mov ecx, [N]

label:
    mov [N], ecx
    mov eax, [N]
    call iprintLF
    loop label

    call quit

```

Рис. 2.2: Код программы lab8-1.asm

После чего создали исполняемый файл и запустили его (рис. 2.3).

```

mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1

```

Рис. 2.3: Запуск программы lab8-1

Далее изменяем содержание lab8-1.asm для демонстрации некорректной работы программы при использовании регистра ecx (рис. 2.4).

```

label:
    sub ecx, 1
    mov [N], ecx
    mov eax, [N]
    call iprintLF
    loop label

```

Рис. 2.4: Измененная часть кода программы lab8-1.asm

Так же создали исполняемый файл и запустили его. В данной программе ecx принимает все значения, но до eax доходит половина, поэтому и число проходов сокращается вдвое (причем округление при получении десятичного числа в большую сторону), что мы видим при выводе, так как ecx за ход цикла убывает на 1 два раза (при sub и loop) (рис. 2.5).



```
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
3
1
```

Рис. 2.5: Запуск измененной программы lab8-1

Затем снова изменили код программы lab8-1.asm, чтобы использовать регистр ecx и при этом сохранить корректную работу цикла, используя стек (рис. 2.6).

```
label:
    push ecx
    sub ecx, 1
    mov [N], ecx
    mov eax, [N]
    call iprintLF
    pop ecx
    loop label
```

Рис. 2.6: Повторно измененная часть кода программы lab8-1.asm

Создали исполняемый файл и запустили его. В результате получили соответствующее число проходов (рис. 2.7).

```
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
4
3
2
1
0
```

Рис. 2.7: Запуск повторно измененной программы lab8-1

Создали файл lab8-2.asm и заполнили его текстом из Листинга 8.2. после внимательного изучения (рис. 2.8 и рис. 2.9).

```
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ gedit lab8-2.asm
```

Рис. 2.8: Создание файла lab8-2.asm

```

#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

    pop ecx

    pop edx

    sub ecx, 1

next:
    cmp ecx, 0
    jz _end

    pop eax
    call sprintLF
    loop next

_end:
    call quit

```

Рис. 2.9: Код программы lab8-2.asm

Далее создали исполняемый файл и запустили его, указав аргументы. В итоге программой было выведено все три аргумента, следовательно, три аргумента было обработано (рис. 2.10).

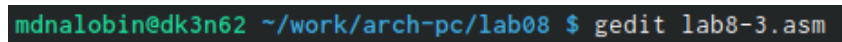
```

mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-2
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-2 1 2 'Листинг 2'
1
2
Листинг 2

```

Рис. 2.10: Запуск программы lab8-2

Создали файл lab8-3.asm и ввели в него текст Листинга 8.3. (рис. 2.11 и рис. 2.12).

A terminal window with a dark background. The prompt is 'mdnalobin@dk3n62' in green. The current directory is '~/work/arch-pc/lab08' in blue. The command '\$ gedit lab8-3.asm' is entered in blue. The terminal output is not visible.

```
mdnalobin@dk3n62 ~/work/arch-pc/lab08 $ gedit lab8-3.asm
```

Рис. 2.11: Создание файла lab8-3.lst

```

#include 'in_out.asm'

SECTION .data
msg: DB 'Результат: ',0

SECTION .text
GLOBAL _start
_start:
    pop ecx

    pop edx

    sub ecx,1

    mov esi, 0

next:
    cmp ecx,0
    jz _end

    pop eax
    call atoi
    add esi,eax

    loop next

_end:
    mov eax,msg
    call sprint
    mov eax,esi
    call iprintLF
    call quit

```

Рис. 2.12: Код программы lab8-3.asm

Создали исполняемый файл и запустили его, указав приведенные аргументы для проверки (рис. 2.13).

```
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-3
Результат: 0
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рис. 2.13: Запуск программы lab8-3

На этом этапе редактируем текст программы таким образом, чтобы она выводила произведение аргументов, создаем исполняемый файл и запускаем его, проверяя на работу при аргументах и их отсутствии. (рис. 2.14 и рис. 2.15).

```

_start:
    pop ecx

    pop edx

    sub ecx,1

    mov esi,1

    cmp ecx,0
    jz _o

next:
    cmp ecx,0
    jz _end

    pop eax
    call atoi
    mul esi
    mov esi,eax

    loop next

_end:
    mov eax,msg
    call sprint
    mov eax,esi
    call iprintLF
    call quit

_o:
    mov esi,0
    jmp _end

```

Рис. 2.14: Код измененной программы lab8-3.asm

```
mdnlobin@dk3n62 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
mdnlobin@dk3n62 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
mdnlobin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-3 12 1 2
Результат: 24
mdnlobin@dk3n62 ~/work/arch-pc/lab08 $ ./lab8-3
Результат: 0
```

Рис. 2.15: Запуск измененной программы lab8-3

### ##Самостоятельная работа

Создали файл sr.asm и написали в нем программу, которая находит сумму значений для некоторой функции, в моем случае -  $4x-3$ , при заданных аргументах. (рис. 2.16 и рис. 2.17).

```
mdnlobin@dk3n62 ~/work/arch-pc/lab08 $ gedit sr.asm
```

Рис. 2.16: Создание файла sr.asm



```

#include 'in_out.asm'

SECTION .data
msg1: DB 'Функция: f(x)=4x-3',0
msg2: DB 'Значения функции: ',0
msg3: DB 'Результат: ',0

SECTION .text
GLOBAL _start
_start:
    mov eax,msg1
    call sprintLF

    pop ecx

    pop edx

    sub ecx,1

    mov esi,0

    mov eax,msg2
    call sprintLF

next:
    cmp ecx,0
    jz _end

    pop eax
    call atoi
    mov ebx, 4
    mul ebx
    dec ebx
    sub eax,ebx
    call iprintLF
    add esi, eax

    loop next

_end:
    mov eax,msg3
    call sprint
    mov eax,esi
    call iprintLF
    call quit

```

Рис. 2.17: Код программы sr.asm

После создали исполняемый файл `sr` и проверили на корректность, используя два набора аргументов (рис. 2.18).

```
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ nasm -f elf sr.asm
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o sr sr.o
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ./sr 1 2 3 4
Функция:  $f(x)=4x-3$ 
Значения функции:
1
5
9
13
Результат: 28
mdnalogin@dk3n62 ~/work/arch-pc/lab08 $ ./sr 1 1 5
Функция:  $f(x)=4x-3$ 
Значения функции:
1
1
17
Результат: 19
```

Рис. 2.18: Запуск программы `sr`

## 3 Выводы

В ходе данной лабораторной работы приобрели навык использования циклов и обработки аргументов командной строки.

...