

Отчёта по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Виме Давид Тененте

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
3.1	Реализация циклов в NASM	5
3.2	Обработка аргументов командной строки.	8
3.3	Задание для самостоятельной работы	10
4	Выводы	13

Список иллюстраций

3.1 Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
3.2 Заполняем файл	7
3.3 Запускаем файл и проверяем его работу	7
3.4 Изменяем файл	8
3.5 Запускаем файл и смотрим на его работу	8
3.6 Редактируем файл	9
3.7 Проверяем, сошелся ли наш вывод с данным в условии вывода .	9
3.8 Создаем файл командой <code>touch</code>	9
3.9 Заполняем файл	10
3.10 Смотрим на работу программ	10
3.11 Создаем файл командой <code>touch</code>	10
3.12 Заполняем файл	11
3.13 Смотрим на работу программы	11
3.14 Изменяем файл.	12
3.15 Проверяем работу файла(работает правильно)	12
3.16 Создаем файл командой <code>touch</code>	13
3.17 Пишем программу	13
3.18 Смотрим на работу программы при $x_1=5$ $x_2=3$ $x_1=4$ (всё верно) . .	13
3.19 Смотрим на работу программы при $x_1=1$ $x_2=3$ $x_1=7$ (всё верно) . .	14

1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. 3.1).

```
lucas@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
lucas@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.1: Создаем каталог с помощью команды mkdir и файл с помощью команды touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. 3.2).

```
lab8-1.asm  [----]  0 L: [ 1+ 0  1/ 24] *(0  / 299b) 0037 0x025  [*] [X]
%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. 3.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. 3.3).

```

lucas@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lucas@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
lucas@fedora:~/work/arch-pc/lab08$

```

Рис. 3.3: Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. 3.4).

```

lab8-1.asm  [-----]  0 L: [ 1+ 0  1/ 24] *(0  / 298b) 0037 0x025  [*][X]
%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:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label

```

Рис. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. 3.5).

```
lucas@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lucas@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
Segmentation fault (core dumped)
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Регистр `ecx` принимает значения 9,7,5,3,1(на вход подается число 10, в цикле `label` данный регистр уменьшается на 2 командой `sub` и `loop`).

Число проходов цикла не соответствует числу `N`, так как уменьшается на 2.

Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис. 3.6).

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

Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. 3.7).

```
lucas@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lucas@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
Segmentation fault (core dumped)
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом В

данном случае число проходов цикла равна числу N.

3.2 Обработка аргументов командной строки.

Создаем новый файл (рис. 3.8).

```
lucas@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. 3.9).

```
lab8-2.asm [----] 0 L: [ 1+ 0 1/ 16] *(0 / 164b) 0037 0x025
#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 sprintf
loop next
_end:
call quit
```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. 3.10).


```
lucas@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
lucas@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.10: Смотрим на работу программ

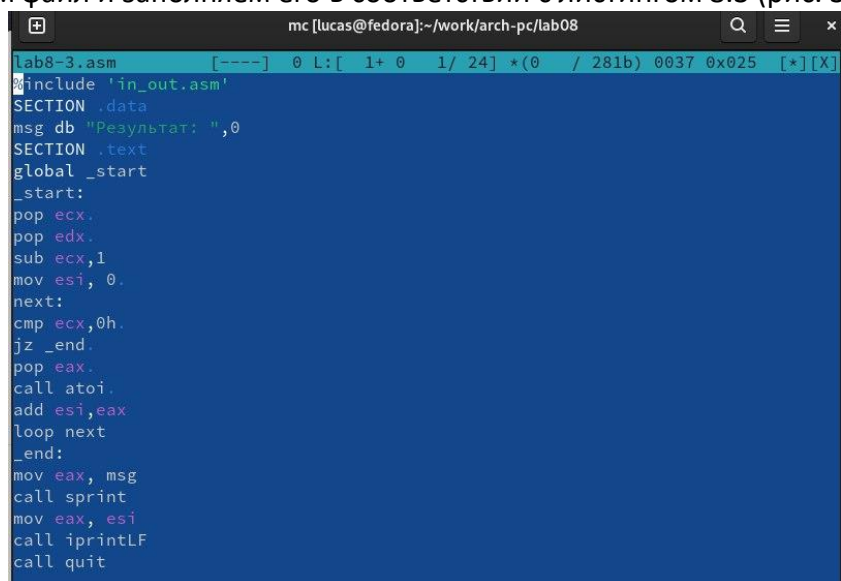
Программой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm (рис. 3.11).

```
lucas@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.11: Создаем файл командой touch

Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. 3.12).



```
lab8-3.asm [----] 0 L: [ 1+ 0 1/ 24] *(0 / 281b) 0037 0x025 [*][X]
#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,0h.
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. 3.12: Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. 3.13).

```

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

```

Рис. 3.13: Смотрим на работу программы

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. 3.14).

```

next:
cmp ecx,0h.
jz _end.
pop eax.
call atoi.
mul esi
mov esi,eax
loop next
_end:

```

Рис. 3.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. 3.15).

```

lucas@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
lucas@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 3 4
Результат: 0
lucas@fedora:~/work/arch-pc/lab08$ mc
lucas@fedora:~/work/arch-pc/lab08$

```

Рис. 3.15: Проверяем работу файла(работает правильно)

3.3 Задание для самостоятельной работы

ВАРИАНТ-20

1. Напишите программу, которая находит сумму значений функции $f(x)$ для x от 1 до n .

= a_1, a_2, \dots, a_n т.е. программа должна вывести значение $(a_1 + a_2) + \dots + a_n$.
 Значения a_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 a_1, a_2, \dots, a_n

Создаем новый файл (рис. 3.16).

```
lucas@fedora:~/work/arch-pc/lab08$ touch lab8-4.asm
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.16: Создаем файл командой touch

Открываем его и пишем программу, которая выведет сумму значений, получившихся
 после решения выражения $3(10+x)$ (рис. 3.17).

```
mc [lucas@fedora]:~/work/arch-pc/lab08
lab8-4.asm [----] 0 L: [ 1+ 0 1/ 30] *(0 / 403b) 0037 0x025 [*]
%include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 12x - 7", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintf
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 12
mul ebx
sub eax, 7
add esi, eax
loop next
_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintLF
call quit
```

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. 3.18).

```
lucas@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
lucas@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-4 5 3 4
Функция:  $f(x) = 12x - 7$ 
Результат: 123
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.18: Смотрим на работу программы при $x_1=5$ $x_2=3$ $x_3=4$ (всё верно)

Транслируем файл и смотрим на работу программы (рис. 3.19).

```
lucas@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 3 7
Функция:  $f(x) = 12x - 7$ 
Результат: 111
lucas@fedora:~/work/arch-pc/lab08$
```

Рис. 3.19: Смотрим на работу программы при $x_1=1$ $x_2=3$ $x_3=7$ (всё верно)

4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.