

Отчёта по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Павлюченков Сергей Витальевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	18
	Список литературы	19

Список иллюстраций

3.1	Создание первого файла	7
3.2	Программа из листинга 8.1	8
3.3	Запуск исполнительного файла	8
3.4	Измененная программа	9
3.5	Запуск исполнительного файла	9
3.6	Измененная программа	10
3.7	Запуск исполнительного файла	11
3.8	Программа из листинга 8.2	12
3.9	Запуск исполнительного файла	12
3.10	Программа из листинга 8.3	13
3.11	Запуск исполнительного файла	14
3.12	Измененная программа	15
3.13	Запуск исполнительного файла	15
3.14	Получившаяся программа	16
3.15	Запуск исполнительного файла	17

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

3 Выполнение лабораторной работы

- Реализация циклов в NASM

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm (рис. 3.1).

```
(svpavliuchenkov@kali)-[~]  
$ mkdir ~/work/arch-pc/lab08/  
  
(svpavliuchenkov@kali)-[~]  
$ cd ~/work/arch-pc/lab08/  
  
(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]  
$ touch lab8-1.asm  
  
(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]  
$
```

Рис. 3.1: Создание первого файла

Ввожу в файл lab8-1.asm текст программы из листинга 8.1 (рис. 3.2).

```
~/work/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
[Icons] [Zoom] [Find] [Run]

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:; ----- Вывод сообщения 'Введите N: '
9 mov eax,msg1
10 call sprint; ----- Ввод 'N'
11 mov ecx, N
12 mov edx, 10
13 call sread; ----- Преобразование 'N' из символа в число
14 mov eax,N
15 call atoi
16 mov [N],eax; ----- Организация цикла
17 mov ecx,[N] ; Счетчик цикла, `ecx-N`
18 label:
19 push ecx ; добавление значения ecx в стек
20 sub ecx,1
21 mov [N],ecx
22 mov eax,[N]
23 call iprintLF
24 pop ecx ; извлечение значения ecx из стека
25 loop label
26 call quit
27
```

Рис. 3.2: Программа из листинга 8.1

Создаю исполняемый файл и проверяю его работу (рис. 3.3).

```
(sypavliuchenkov@kali) - [~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
(sypavliuchenkov@kali) - [~/work/arch-pc/lab08]
$
```

Рис. 3.3: Запуск исполняемого файла

Этот алгоритм выводит все целые числа от N до 1.

Изменяю текст программы добавив изменение значение регистра ecx в цикле (рис. 3.3).

```
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:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'; переход на 'label'
call quit
```

Рис. 3.4: Измененная программа

Создаю исполняемый файл и проверяю его работу (рис. 3.5).

```
(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-1.asm

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-1 lab8-1.o

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 10
9
7
5
3
1

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$
```

Рис. 3.5: Запуск исполняемого файла

Число проходов цикла не соответствует значению ☒ введенному с клавиатуры.

Вношу изменения в текст программы добавив команду push и pop для сохранения значения счетчика цикла loop (рис. 3.6).

```
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 push ecx ; добавление значения ecx в стек
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 pop ecx ; извлечение значения ecx из стека
29 loop label
30
```

Рис. 3.6: Измененная программа

Создаю исполняемый файл и проверяю его работу (рис. 3.5).

```
(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-1.asm

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-1 lab8-1.o

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$
```

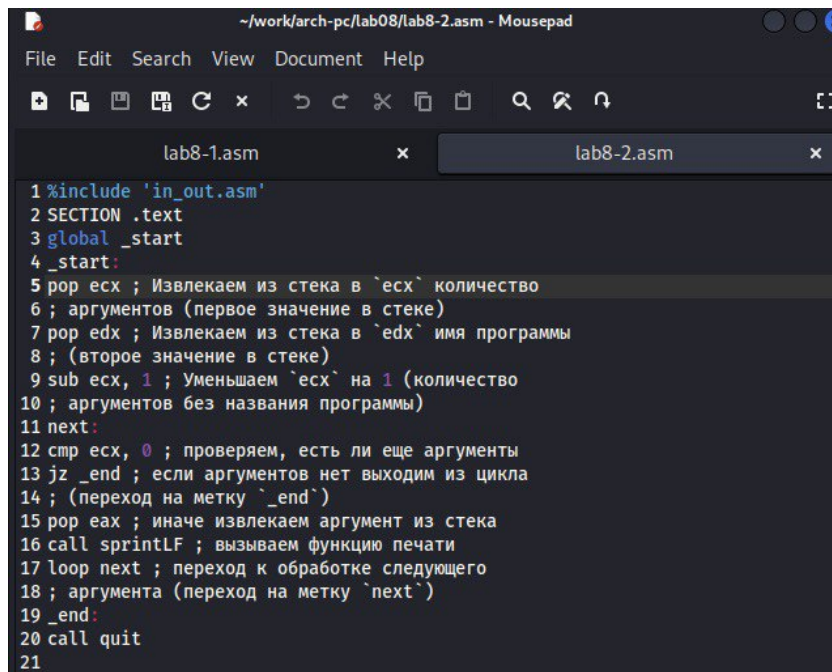
Рис. 3.7: Запуск исполнительного файла

Программа выводит все целые числа от N-1 до 0 (рис. 3.7).

Число проходов цикла соответствует значению ☒ введенному с клавиатуры.

- Обработка аргументов командной строки

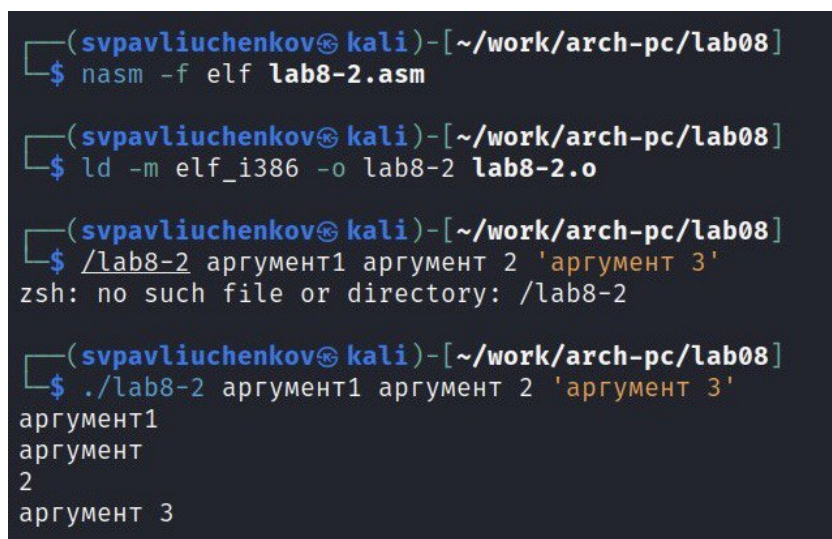
Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы из листинга 8.2 (рис. 3.8).

A screenshot of a text editor window titled '~/.work/arch-pc/lab08/lab8-2.asm - Mousepad'. The window has a menu bar with 'File', 'Edit', 'Search', 'View', 'Document', and 'Help'. Below the menu bar is a toolbar with various icons. There are two tabs open: 'lab8-1.asm' and 'lab8-2.asm'. The 'lab8-2.asm' tab is active, showing the following assembly code:

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

Рис. 3.8: Программа из листинга 8.2

Создаю исполняемый файл и проверяю его работу (рис. 3.9).

A screenshot of a terminal window with the prompt '(svpavliuchenkov@kali) - [~/work/arch-pc/lab08]'. The user enters the following commands and receives the following output:

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

(svpavliuchenkov@kali) - [~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-2 lab8-2.o

(svpavliuchenkov@kali) - [~/work/arch-pc/lab08]
$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
zsh: no such file or directory: ./lab8-2

(svpavliuchenkov@kali) - [~/work/arch-pc/lab08]
$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 3.9: Запуск исполняемого файла

Программой было обработано 4 аргумента, так как 2 не было взято в кавычки, из-за чего засчиталось за отдельный аргумент.

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.

Создаю файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввожу в него текст программы из листинга 8.3(рис. 3.10).

```
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 push ecx ; добавление значения ecx в стек
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 pop ecx ; извлечение значения ecx из стека
29 loop label
30
```

Рис. 3.10: Программа из листинга 8.3

Создаю исполняемый файл и проверяю его работу (рис. 3.11).

```
(svpavliuchenkov@kali)-[~/Downloads]
$ nasm -f elf lab8-2.asm

(svpavliuchenkov@kali)-[~/Downloads]
$ ld -m _i386 -o lab8-2 lab8-2.o
ld: unrecognised emulation mode: _i386
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 e

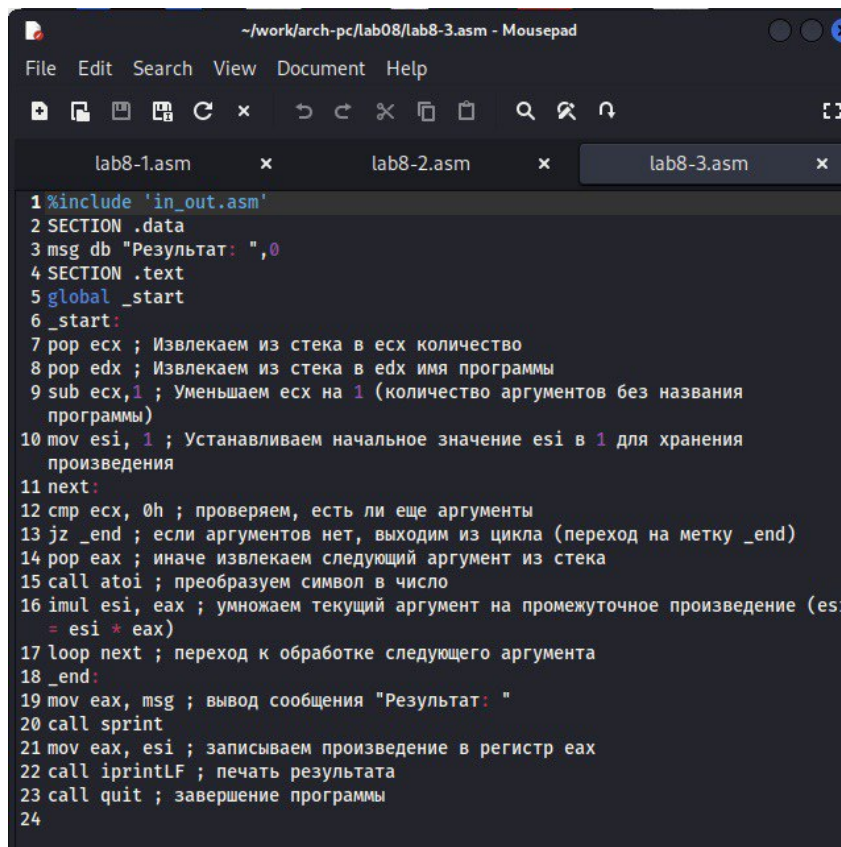
(svpavliuchenkov@kali)-[~/Downloads]
$ ld -m elf_i386 -o lab8-2 lab8-2.o

(svpavliuchenkov@kali)-[~/Downloads]
$ ./lab8-2 3 4 5
Результат: 12

(svpavliuchenkov@kali)-[~/Downloads]
$
```

Рис. 3.11: Запуск исполнительного файла

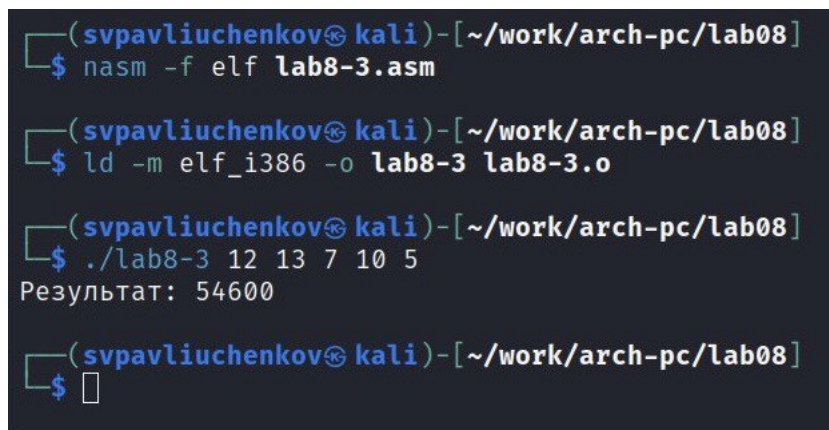
Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки(рис. 3.12)



```
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 pop edx ; Извлекаем из стека в edx имя программы
9 sub ecx,1 ; Уменьшаем ecx на 1 (количество аргументов без названия
    программы)
10 mov esi, 1 ; Устанавливаем начальное значение esi в 1 для хранения
    произведения
11 next:
12 cmp ecx, 0h ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет, выходим из цикла (переход на метку _end)
14 pop eax ; иначе извлекаем следующий аргумент из стека
15 call atoi ; преобразуем символ в число
16 imul esi, eax ; умножаем текущий аргумент на промежуточное произведение (esi
    = esi * eax)
17 loop next ; переход к обработке следующего аргумента
18 _end:
19 mov eax, msg ; вывод сообщения "Результат: "
20 call sprint
21 mov eax, esi ; записываем произведение в регистр eax
22 call iprintf ; печать результата
23 call quit ; завершение программы
24
```

Рис. 3.12: Измененная программа

Создаю исполняемый файл и проверяю его работу (рис. 3.3).



```
(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-3.asm

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-3 lab8-3.o

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ./lab8-3 12 13 7 10 5
Результат: 54600

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$
```

Рис. 3.13: Запуск исполнительного файла

- Задание для самостоятельной работы

Пишу программу, которая находит сумму значений функции $(12x - 7)$ 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.

Создаю файл `problem.asm` пишу для него программу (рис. 3.14).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db "Функция 12*x - 7", 0
4 msg2 db "Результат: ", 0
5 newline db 10, 0
6 SECTION .text
7 global _start
8 _start:
9 pop ecx ; Извлекаем из стека в ecx количество
10 pop edx ; Извлекаем из стека в edx имя программы
11 sub ecx, 1 ; Уменьшаем ecx на 1 (количество аргументов без названия программы)
12 mov esi, 0 ; Устанавливаем начальное значение esi в 1 для хранения произведения
13 next:
14 cmp ecx, 0h ; проверяем, есть ли еще аргументы
15 jz _end ; если аргументов нет, выходим из цикла (переход на метку _end)
16 pop eax ; иначе извлекаем следующий аргумент из стека
17 call atoi ; преобразуем символ в число
18 mov ebx, 12
19 mul ebx ; умножаем текущий аргумент на промежуточное произведение (esi = esi * eax)
20 add eax, -7;
21 add esi, eax;
22 loop next ; переход к обработке следующего аргумента
23 _end:
24 mov eax, msg1;
25 call sprint;
26 mov eax, newline
27 call sprint
28 mov eax, msg2 ; вывод сообщения "Результат: "
29 call sprint;
30 mov eax, esi ; записываем произведение в регистр eax
31 call iprintLF ; печать результата
32 call quit ; завершение программы
33

```

Рис. 3.14: Получившаяся программа

Создаю исполняемый файл и проверяю его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.


```
(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ nasm -f elf problem.asm

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o problem problem.o

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ./problem 1 2 3 4
Функция  $12 \cdot x - 7$ 
Результат: 92

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ./problem 1 2 3
Функция  $12 \cdot x - 7$ 
Результат: 51

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$ ./problem 1 2
Функция  $12 \cdot x - 7$ 
Результат: 22

(svpavliuchenkov@kali)-[~/work/arch-pc/lab08]
$
```

Рис. 3.15: Запуск исполнительного файла

Программа выдает точные значения.

4 Выводы

Я научился писать программы с использованием циклов и обработкой аргументов командной строки.

Список литературы

Лабораторная работа №8

GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>

NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>