

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ
НАРОДОВ

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

**ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8**

дисциплина: Архитектура компьютера

Студент: Силантьева Анастасия

Группа: НКАбд-05-25

№ ст. билета: 1032253541

МОСКВА

2025 г.

Содержание

Список иллюстраций	3
1. Цель работы	4
2. Выполнение лабораторной работы	5
2.1 Реализация циклов в NASM	5
2.2 Обработка аргументов командной строки	8
3. Задание для самостоятельной работы	12

Список иллюстраций

1. Рис. 1 Создание каталога
2. Рис. 2 Копирование программы из листинга
3. Рис. 3 Запуск программы
4. Рис. 4 Изменение программы
5. Рис. 5 Запуск измененной программы
6. Рис. 6 Добавление push и pop в цикл программы
7. Рис. 7 Запуск измененной программы
8. Рис. 8 Копирование программы из листинга
9. Рис. 9 Запуск второй программы
10. Рис. 10 Копирование программы из третьего листинга
11. Рис. 11 Запуск третьей программы
12. Рис. 12 Изменение третьей программы
13. Рис. 13 Запуск измененной третьей программы
14. Рис. 14 Написание программы для самостоятельной работы
15. Рис. 15 Запуск программы для самостоятельной работы

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

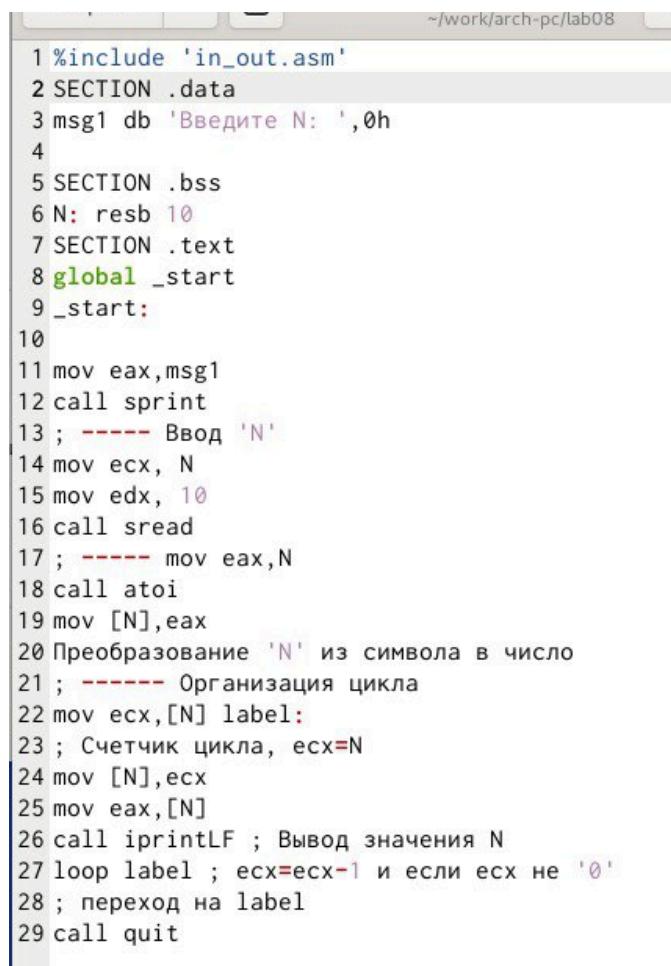
Создаю каталог для программ лабораторной работы №8. ([Рис. 1](#))



```
azsilantjeva@dk6n13 ~ $ mkdir ~/work/arch-pc/lab08
azsilantjeva@dk6n13 ~ $ cd ~/work/arch-pc/lab08
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ touch lab8-1.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 1 Создание каталога

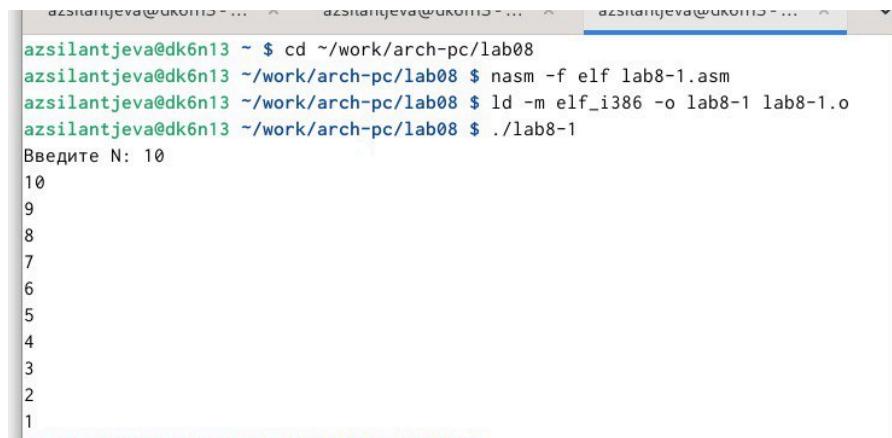
Копирую в созданный файл программу из листинга. ([Рис. 2](#))



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

Рис. 2 Копирование программы из листинга

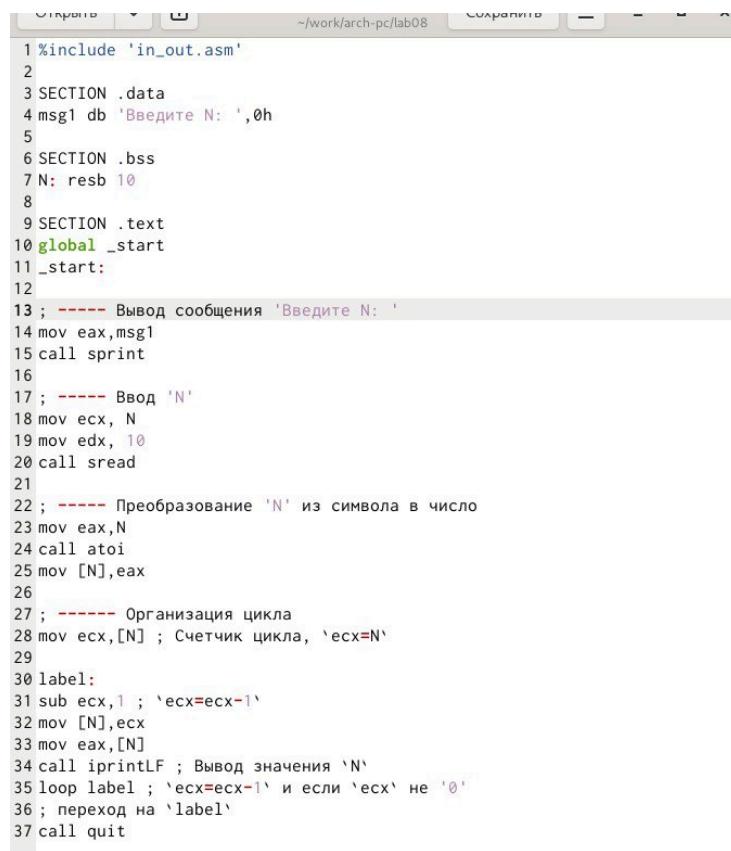
Запускаю программу, она показывает работу циклов в NASM. [\(рис. 3\)](#).



```
azsilantjeva@dk6n13 ~ $ cd ~/work/arch-pc/lab08
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
```

Рис. 3 Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра `ecx`. [\(рис. 4\)](#).



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

Рис. 4 Изменение программы

Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое. [\(рис. 5\)](#).

```
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 5 Запуск измененной программы

Добавляю команды push и pop в программу. [\(рис. 6\)](#)

```
Открыть  в  Сохранить lab8-1.asm
~/work/arch-pc/lab08
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите N: ',0h
5
6 SECTION .bss
7 N: resb 10
8
9 SECTION .text
10 global _start
11 _start:
12
13 ; ----- Вывод сообщения 'Введите N: '
14 mov eax,msg1
15 call sprint
16
17 ; ----- Ввод 'N'
18 mov ecx, N
19 mov edx, 10
20 call sread
21
22 ; ----- Преобразование 'N' из символа в число
23 mov eax,N
24 call atoi
25 mov [N],eax
26
27 ; ----- Организация цикла
28 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
29
30 label:
31 push ecx ; добавление значения ecx в стек
32 sub ecx,1
33 mov [N],ecx
34 mov eax,[N]
35 call iprintf
36 pop ecx ; извлечение значения ecx из стека
37 loop label
38 ; переход на 'label'
39 call quit
```

Рис. 6 Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 ([рис. 7](#)).

```
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 7 Запуск измененной программы

2.2 Обработка аргументов командной строки

Создаю новый рабочий файл и вставляю в него код из следующего листинга ([рис. 8](#)).

```
%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 sprintLF ; вызываем функцию печати
    loop next ; переход к обработке следующего
    ; аргумента (переход на метку `next`)
_end:
    call quit
```

Рис. 8 Копирование программы из листинга

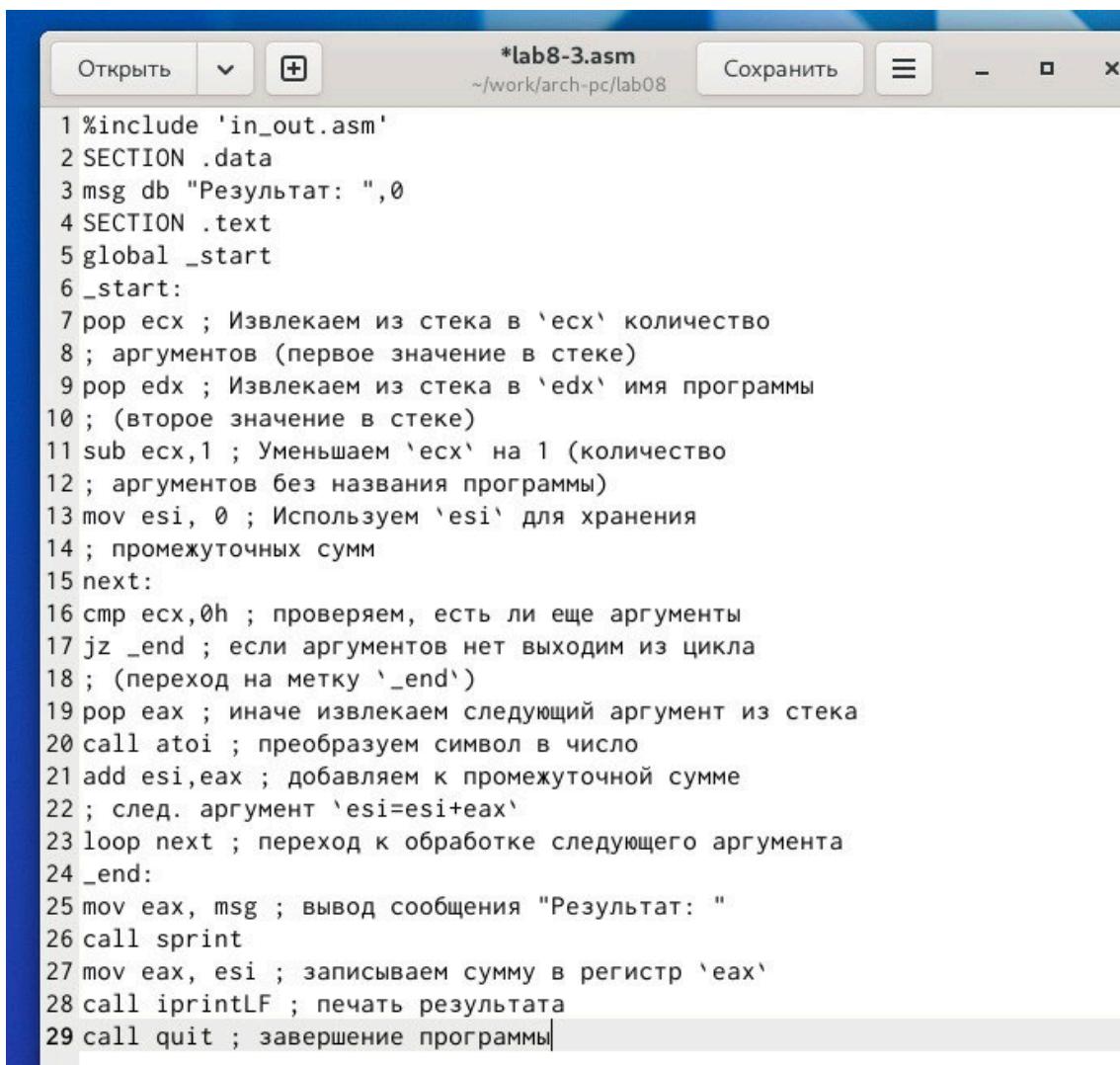
Запускаю вторую программу. ([рис. 9](#))



```
azsilantjeva@dk6n13 ~ $ cd ~/work/arch-pc/lab08
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 9 Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга ([рис. 10](#)).



```
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
```

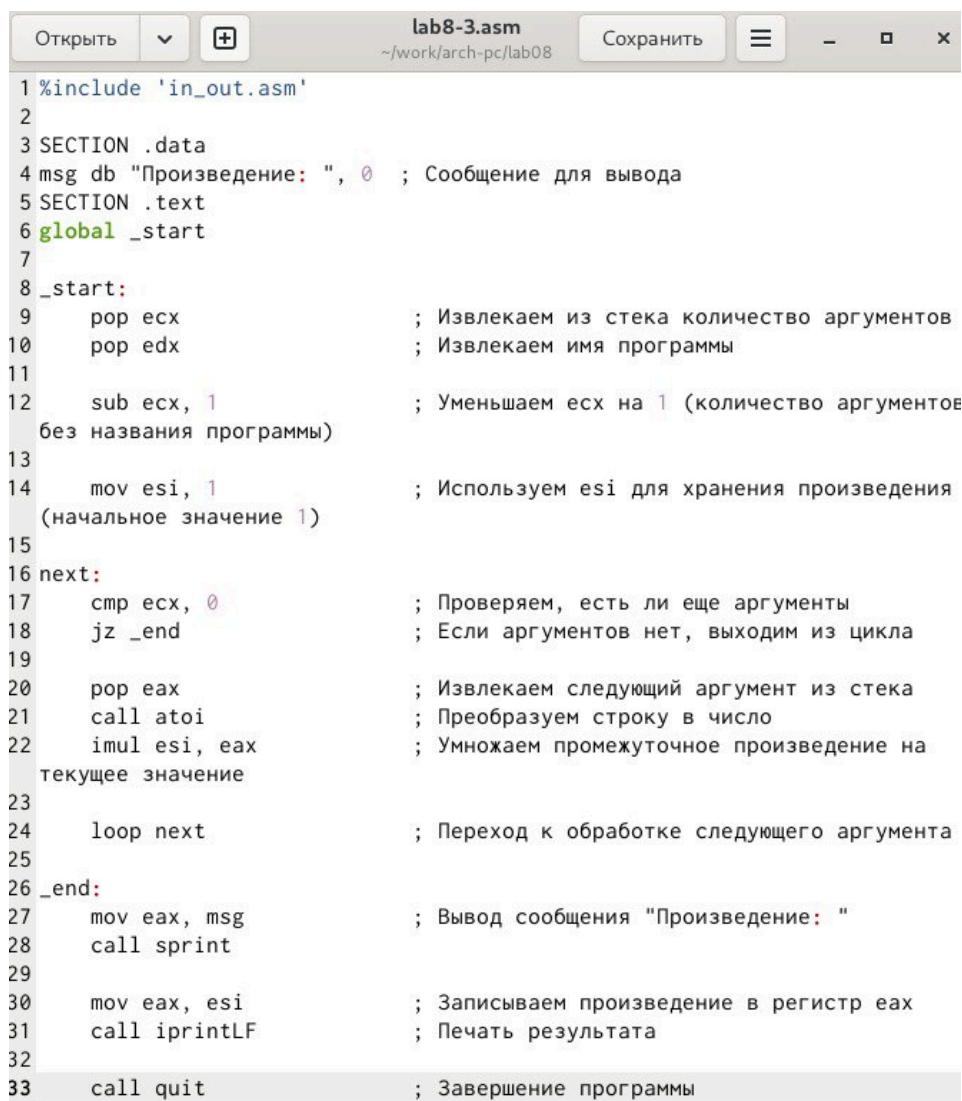
Рис. 10 Копирование программы из третьего листинга

Компилирую программу и запускаю ее, указав в качестве аргументов некоторые числа, которые программа складывает ([рис. 11](#)).

```
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ touch lab8-3.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
из azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 11 Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала ([рис. 12](#)).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Произведение: ", 0 ; Сообщение для вывода
5 SECTION .text
6 global _start
7
8 _start:
9     pop ecx          ; Извлекаем из стека количество аргументов
10    pop edx          ; Извлекаем имя программы
11
12    sub ecx, 1        ; Уменьшаем ecx на 1 (количество аргументов
13    без названия программы)
14    mov esi, 1        ; Используем esi для хранения произведения
15    (начальное значение 1)
16 next:
17    cmp ecx, 0        ; Проверяем, есть ли еще аргументы
18    jz _end            ; Если аргументов нет, выходим из цикла
19
20    pop eax          ; Извлекаем следующий аргумент из стека
21    call atoi          ; Преобразуем строку в число
22    imul esi, eax      ; Умножаем промежуточное произведение на
23    текущее значение
24    loop next          ; Переход к обработке следующего аргумента
25
26 _end:
27    mov eax, msg        ; Вывод сообщения "Произведение: "
28    call sprint
29
30    mov eax, esi        ; Записываем произведение в регистр eax
31    call iprintLF      ; Печать результата
32
33    call quit          ; Завершение программы
```

Рис. 12 Изменение третьей программы

Программа действительно теперь умножает данные на вход числа ([рис. 13](#)).



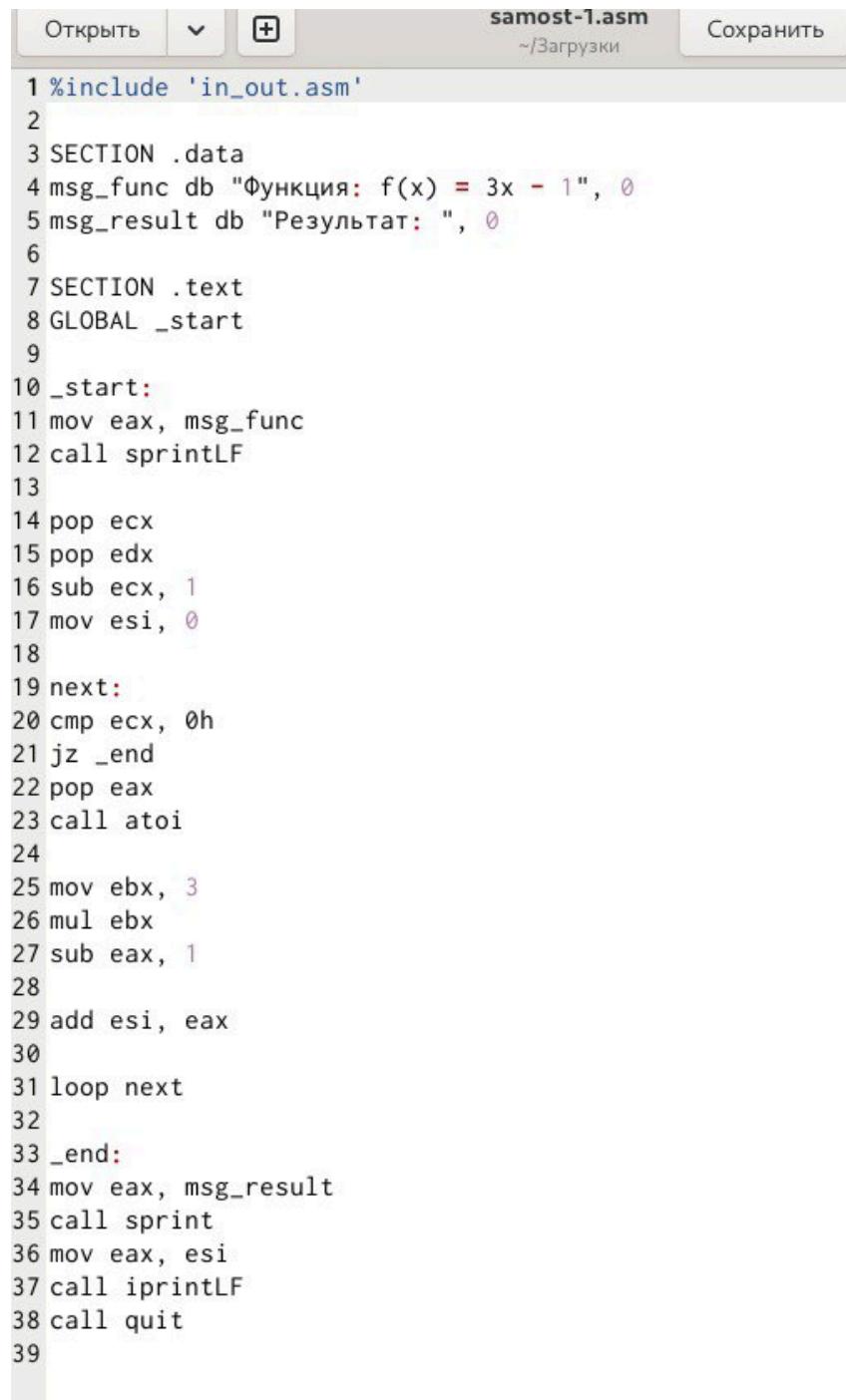
The screenshot shows a terminal window with the title "azsilantjeva@dk6n08 - lab08". The terminal displays the following command-line session:

```
azsilantjeva@dk6n08 ~ $ cd ~/work/arch-pc/lab08
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ./lab8-3 111 1 6
Произведение: 666
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $
```

Рис. 13 Запуск измененной третьей программы

3. Задание для самостоятельной работы

Пишу программу, которая будет находить сумму значений для функции $f(x) = 3x - 1$, которая совпадает с моим 2 вариантом. ([рис.14](#))



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg_func db "Функция: f(x) = 3x - 1", 0
5 msg_result db "Результат: ", 0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11 mov eax, msg_func
12 call sprintLF
13
14 pop ecx
15 pop edx
16 sub ecx, 1
17 mov esi, 0
18
19 next:
20 cmp ecx, 0h
21 jz _end
22 pop eax
23 call atoi
24
25 mov ebx, 3
26 mul ebx
27 sub eax, 1
28
29 add esi, eax
30
31 loop next
32
33 _end:
34 mov eax, msg_result
35 call sprint
36 mov eax, esi
37 call iprintLF
38 call quit
39
```

Рис. 14 Написание программы для самостоятельной работы

Проверяю работу программы, указав в качестве аргумента несколько чисел ([рис. 15](#)).

```
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ touch samost-1.asm
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ nasm -f elf samost-1.asm
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o samost-1 samost-1.o
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ./samost-1123
bash: ./samost-1123: Нет такого файла или каталога
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ./samost-1 1 2 3
Функция: f(x) = 3x - 1
Результат: 15
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $
```

Рис. 15 Запуск программы для самостоятельной работы

3. Вывод

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASMAssembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. —
6. 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
7. RobbinsA. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
8. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
9. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
10. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
11. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).