

# **ОтчеТт по лабораторной работе №8**

**дисциплина: Архитектура компьютера**

Дельгадильо Валерия

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
2.1	Организация стека . . . . .	6
2.2	Инструкции организации циклов . . . . .	6
<b>3</b>	<b>Лабораторной работы</b>	<b>8</b>
3.1	Реализация циклов в NASM . . . . .	8
3.2	Обработка аргументов командной строки . . . . .	13
<b>4</b>	<b>Задание для самостоятельной работы</b>	<b>17</b>
<b>5</b>	<b>Выводы</b>	<b>19</b>
<b>6</b>	<b>Список литературы</b>	<b>20</b>

# Список иллюстраций

3.1	.....	8
3.2	.....	9
3.3	.....	9
3.4	.....	10
3.5	.....	11
3.6	.....	12
3.7	.....	12
3.8	.....	13
3.9	.....	13
3.10	.....	14
3.11	.....	14
3.12	.....	15
3.13	.....	15
3.14	.....	16
4.1	.....	18
4.2	.....	18

## Список таблиц

# 1 Цель работы

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

## 2 Теоретическое введение

### 2.1 Организация стека

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.

Для стека существует две основные операции:

- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop).

### 2.2 Инструкции организации циклов

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее

простой является инструкция loop. Она позволяет организовать безусловный цикл, типичная структура.

Инструкция loop выполняется в два этапа. Сначала из регистра esx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

## 3 Лабораторной работы

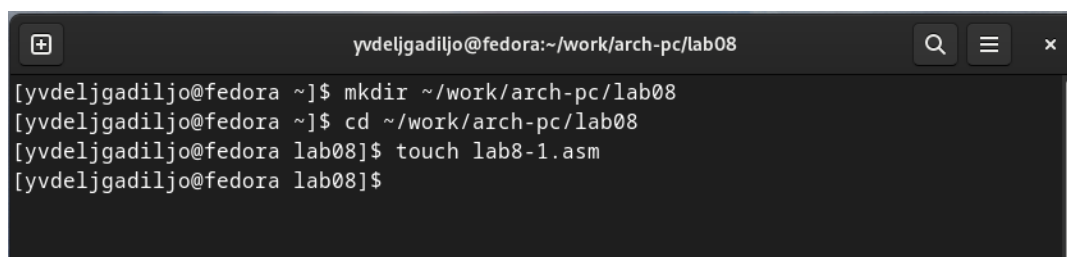
### 3.1 Реализация циклов в NASM

Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm:

```
mkdir ~/work/arch-pc/lab08
```

```
cd ~/work/arch-pc/lab08
```

```
touch lab8-1.asm
```

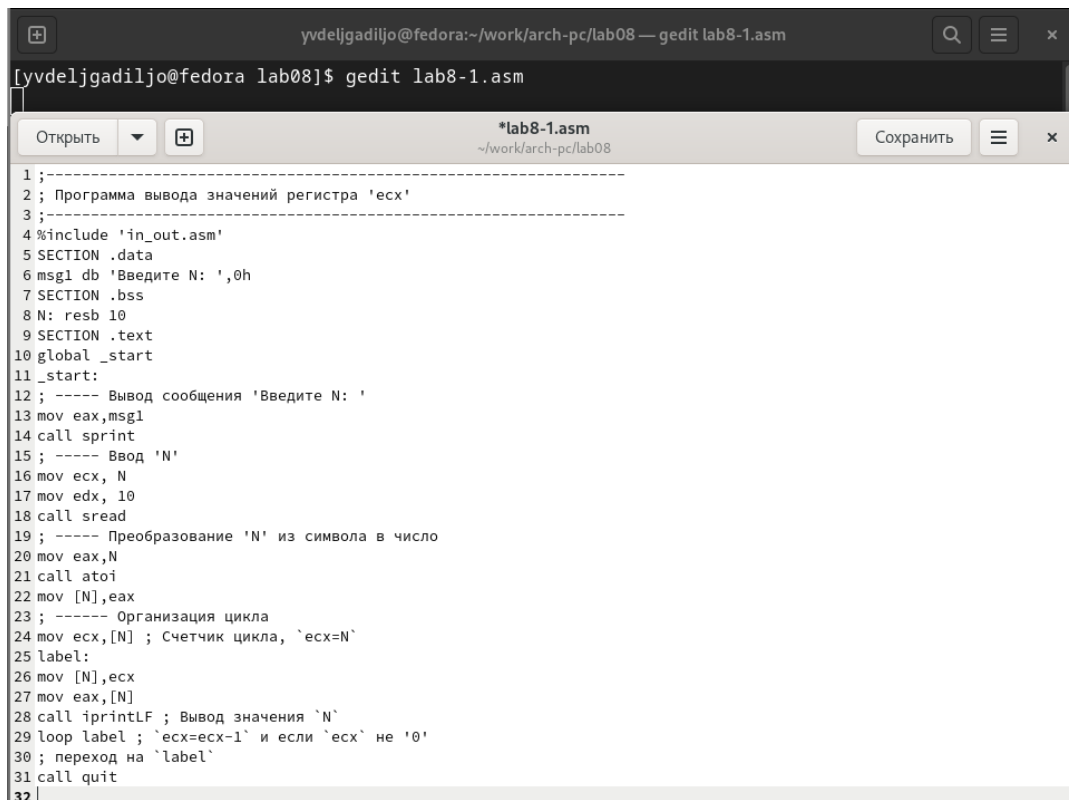
A screenshot of a terminal window with a dark background. The window title is 'yvdeljgadiljo@fedora:~/work/arch-pc/lab08'. The terminal shows the following commands and their outputs: [yvdeljgadiljo@fedora ~]\$ mkdir ~/work/arch-pc/lab08, [yvdeljgadiljo@fedora ~]\$ cd ~/work/arch-pc/lab08, [yvdeljgadiljo@fedora lab08]\$ touch lab8-1.asm, and [yvdeljgadiljo@fedora lab08]\$.

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

Рис. 3.1:

При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр еsx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра еsx. Внимательно изучите текст программы (Листинг 8.1). Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.

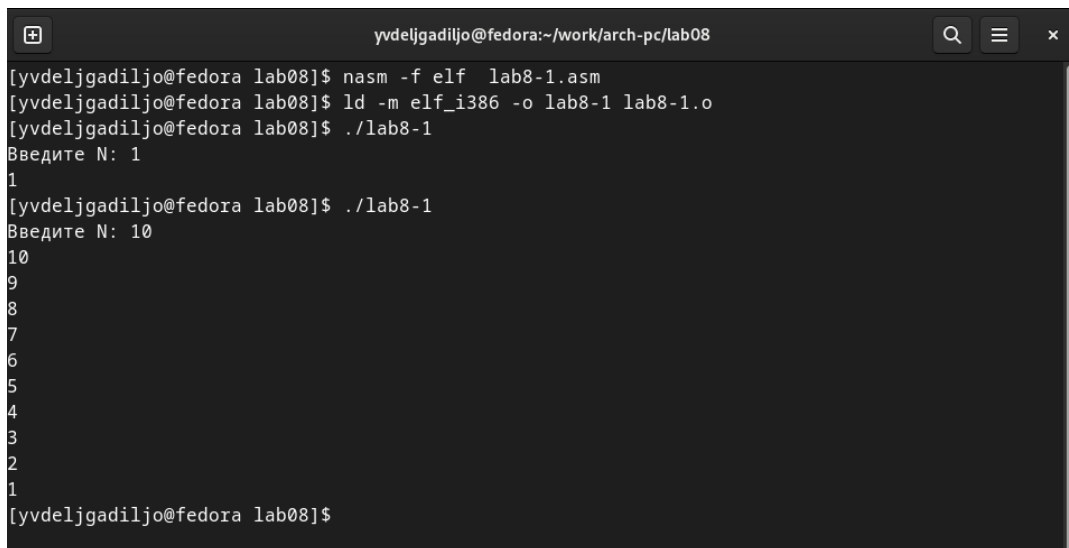




```
[yvdeljgadiljo@fedora lab08]$ gedit lab8-1.asm

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

Рис. 3.2:



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

Рис. 3.3:

Данный пример показывает, что использование регистра ecx в теле цикла loop

может привести к некорректной работе программы. Измените текст программы добавив изменение значение регистра ecx в цикле:label:

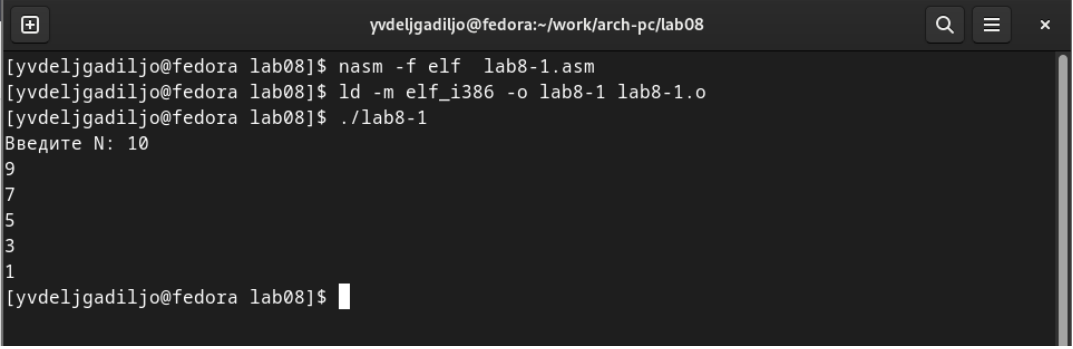
```
sub ecx,1 ; 'ecx=ecx-1'  
mov [N],ecx  
mov eax,[N]  
call iprintLFloop label
```



```
*lab8-1.asm  
~/work/arch-pc/lab08  
Сохранить  
1 ;-----  
2 ; Программа вывода значений регистра 'ecx'  
3 ;-----  
4 %include 'in_out.asm'  
5 SECTION .data  
6 msg1 db 'Введите N: ',0h  
7 SECTION .bss  
8 N: resb 10  
9 SECTION .text  
10 global _start  
11 _start:  
12 ; ---- Вывод сообщения 'Введите N: '  
13 mov eax,msg1  
14 call sprint  
15 ; ---- Ввод 'N'  
16 mov ecx, N  
17 mov edx, 10  
18 call sread  
19 ; ---- Преобразование 'N' из символа в число  
20 mov eax,N  
21 call atoi  
22 mov [N],eax  
23 ; ----- Организация цикла  
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'  
25 label:  
26 sub ecx,1 ; 'ecx=ecx-1'  
27 mov [N],ecx  
28 mov eax,[N]  
29 call iprintLF  
30 loop label  
31 |  
32 call quit  
22
```

Рис. 3.4:

Создайте исполняемый файл и проверьте его работу.

A terminal window titled 'yvdeljgatiljo@fedora:~/work/arch-pc/lab08'. The terminal shows the following commands and output:

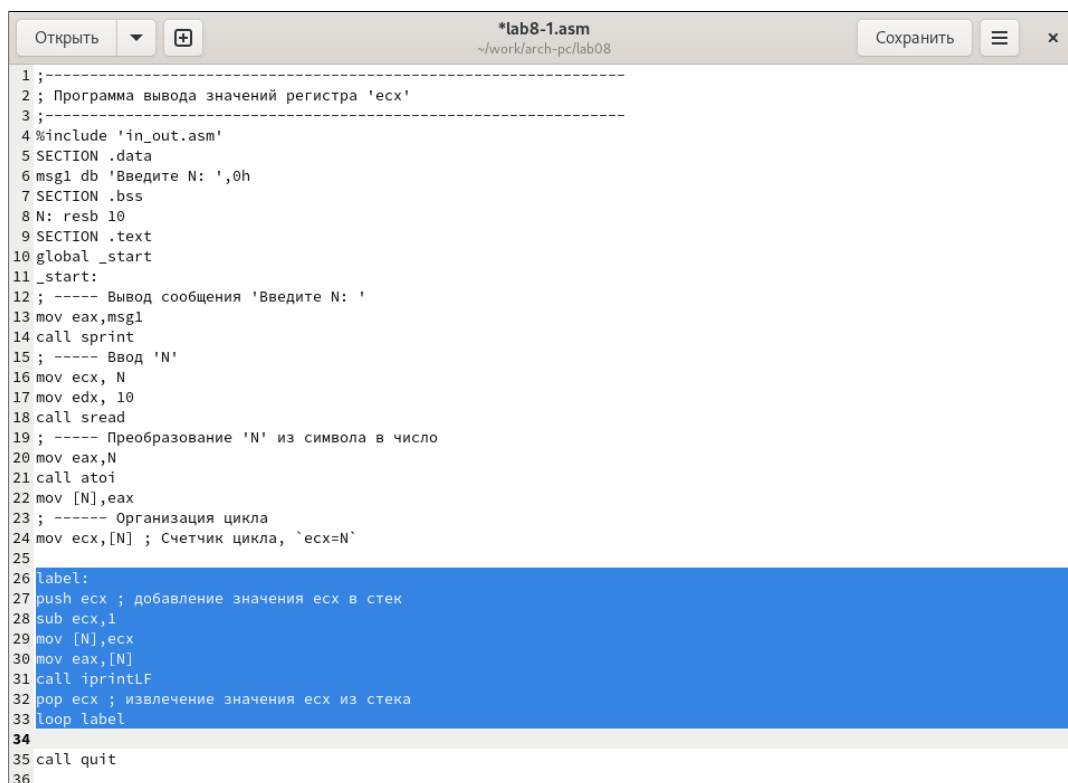
```
[yvdeljgatiljo@fedora lab08]$ nasm -f elf lab8-1.asm
[yvdeljgatiljo@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[yvdeljgatiljo@fedora lab08]$ ./lab8-1
Введите N: 10
9
7
5
3
1
[yvdeljgatiljo@fedora lab08]$
```

Рис. 3.5:

Какие значения принимает регистр `ecx` в цикле? Соответствует ли число проходов цикла значению `N` введенному с клавиатуры?

Нет, в данном случае `N` равно 10, а число проходов цикла равно 5.

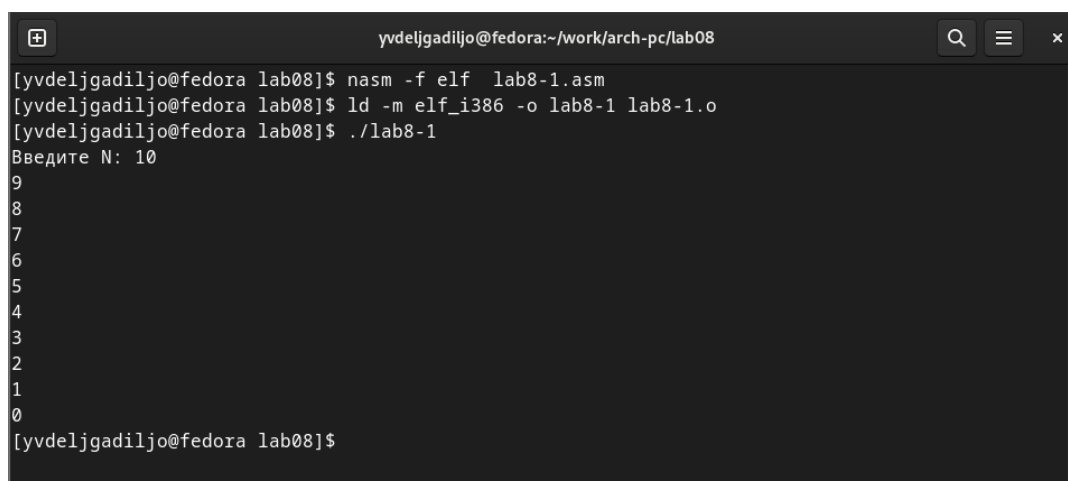
Внесите изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`:



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

Рис. 3.6:

Создайте исполняемый файл и проверьте его работу.



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

Рис. 3.7:

Соответствует ли в данном случае число проходов цикла значению N введен-

ному с клавиатуры? Да.

## 3.2 Обработка аргументов командной строки

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08

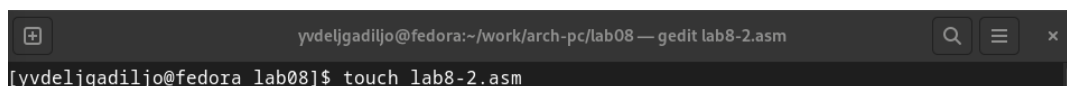


Рис. 3.8:

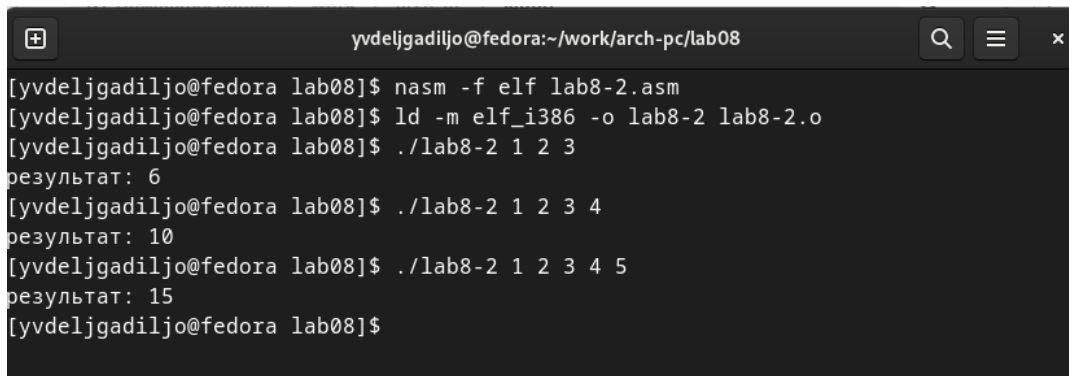
Введите в него текст программы из листинга 8.2.



Рис. 3.9:

Создайте исполняемый файл и запустите его, указав аргументы:

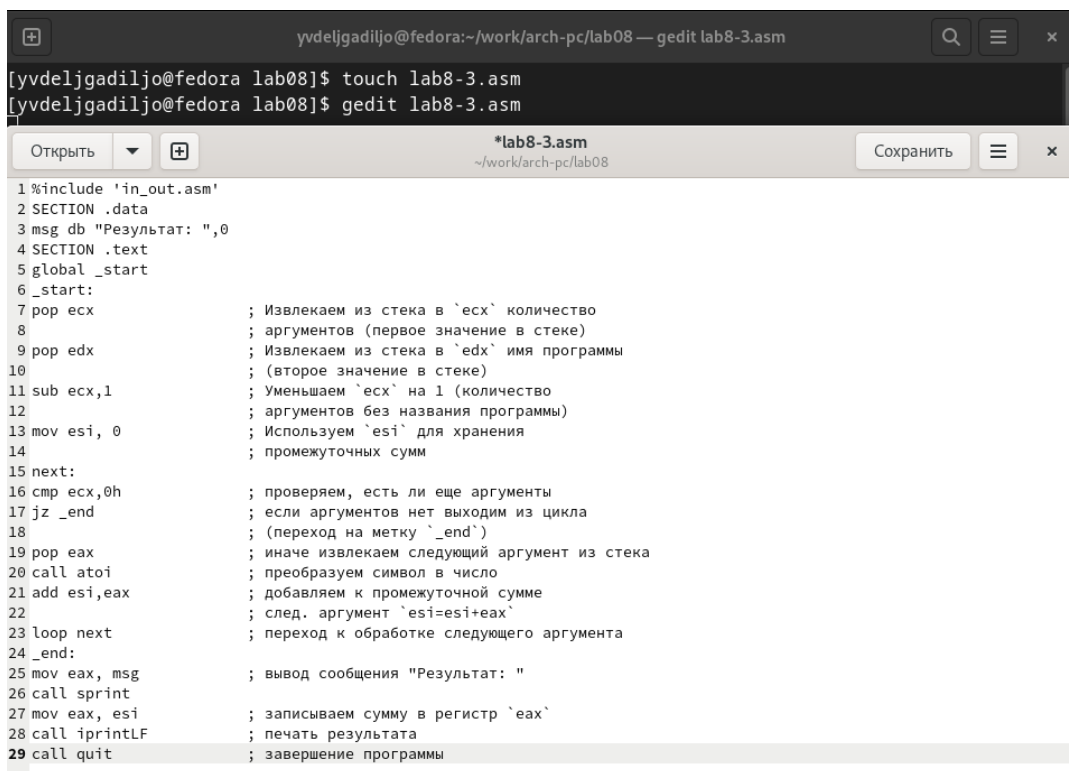
user@dk4n31:~\$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'



```
yvdeljgadiljo@fedora:~/work/arch-pc/lab08
[yvdeljgadiljo@fedora lab08]$ nasm -f elf lab8-2.asm
[yvdeljgadiljo@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[yvdeljgadiljo@fedora lab08]$ ./lab8-2 1 2 3
результат: 6
[yvdeljgadiljo@fedora lab08]$ ./lab8-2 1 2 3 4
результат: 10
[yvdeljgadiljo@fedora lab08]$ ./lab8-2 1 2 3 4 5
результат: 15
[yvdeljgadiljo@fedora lab08]$
```

Рис. 3.10:

Создайте файл lab8-3.asm в каталоге ~/work/archpc/lab08 и введите в него текст программы из листинга 8.3.



```
yvdeljgadiljo@fedora:~/work/arch-pc/lab08 — gedit lab8-3.asm
[yvdeljgadiljo@fedora lab08]$ touch lab8-3.asm
[yvdeljgadiljo@fedora lab08]$ gedit lab8-3.asm

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
```

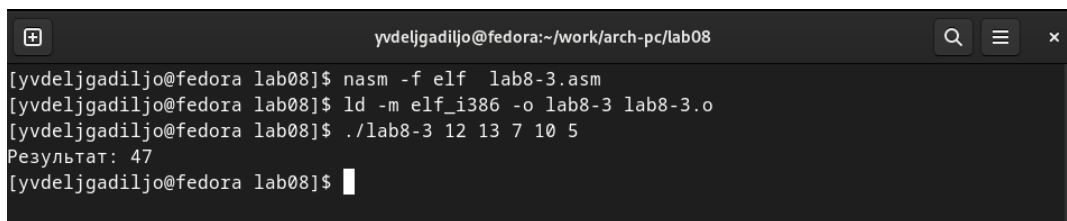
Рис. 3.11:

Создайте исполняемый файл и запустите его, указав аргументы. Пример результата работы программы:

user@dk4n31:~\$ ./main 12 13 7 10 5

Результат: 47

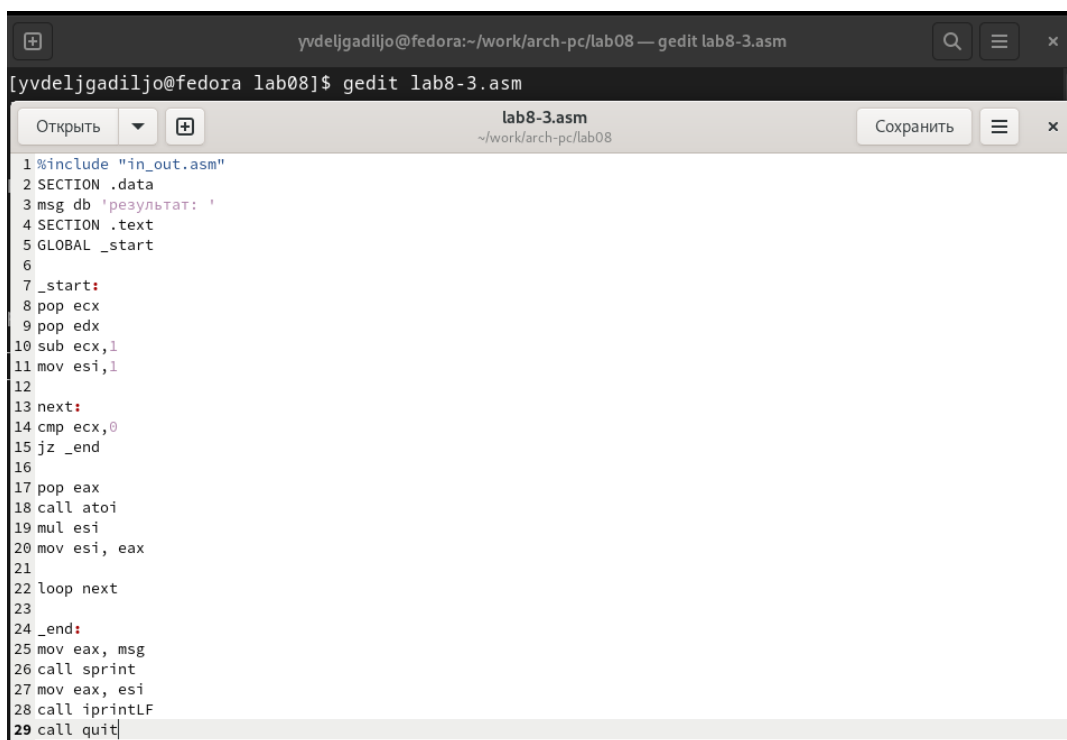
user@dk4n31:~\$



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

Рис. 3.12:

Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.



```
yvdeljgatiljo@fedora:~/work/arch-pc/lab08 — gedit lab8-3.asm
[yvdeljgatiljo@fedora lab08]$ gedit lab8-3.asm

lab8-3.asm
~/work/arch-pc/lab08

1 %include "in_out.asm"
2 SECTION .data
3 msg db 'результат: '
4 SECTION .text
5 GLOBAL _start
6
7 _start:
8 pop ecx
9 pop edx
10 sub ecx,1
11 mov esi,1
12
13 next:
14 cmp ecx,0
15 jz _end
16
17 pop eax
18 call atoi
19 mul esi
20 mov esi, eax
21
22 loop next
23
24 _end:
25 mov eax, msg
26 call sprint
27 mov eax, esi
28 call iprintLF
29 call quit
```

Рис. 3.13:

```
[yvdeljgatiljo@fedora lab08]$ gedit lab8-3.asm  
[yvdeljgatiljo@fedora lab08]$ nasm -f elf lab8-3.asm  
[yvdeljgatiljo@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o  
[yvdeljgatiljo@fedora lab08]$ ./lab8-3 1 2 3 4 5  
результат: 120
```

Рис. 3.14:

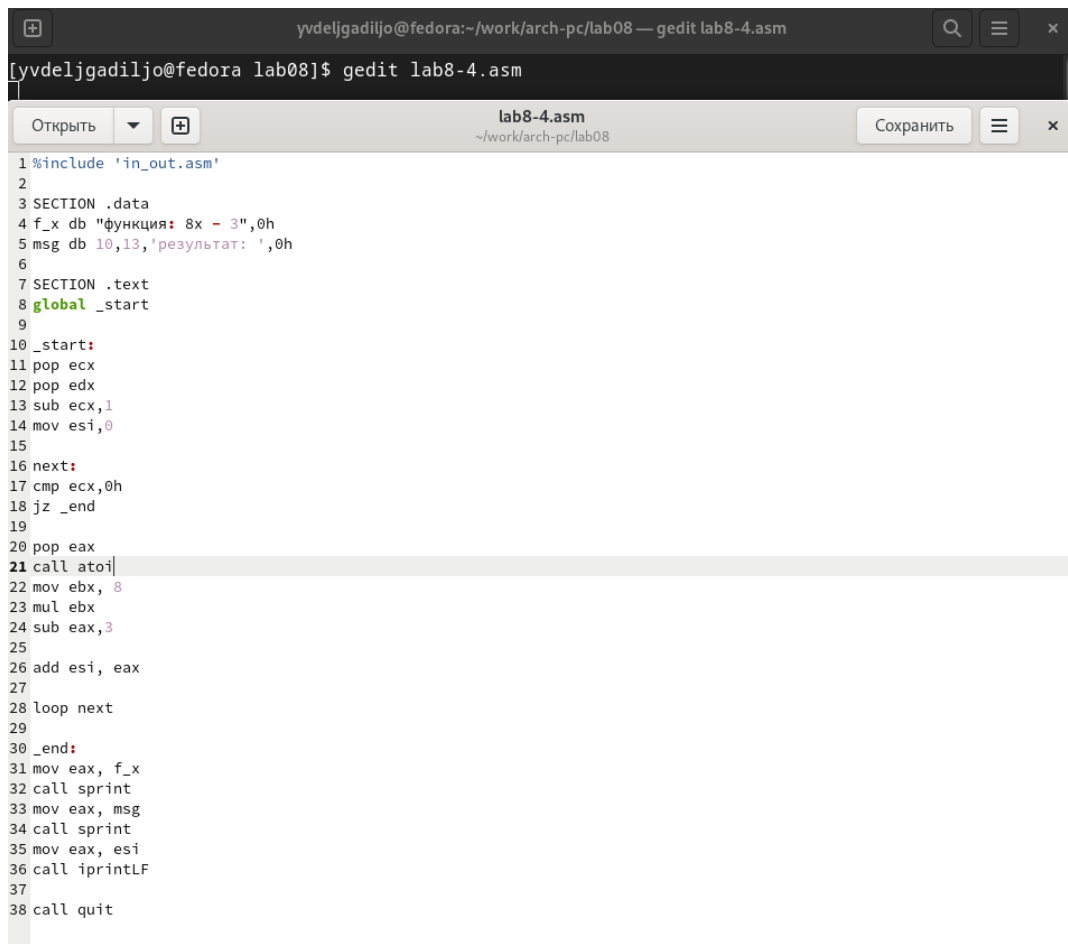


## 4 Задание для самостоятельной работы

Номер варианта: 19

$$F(x) = 8x - 3$$

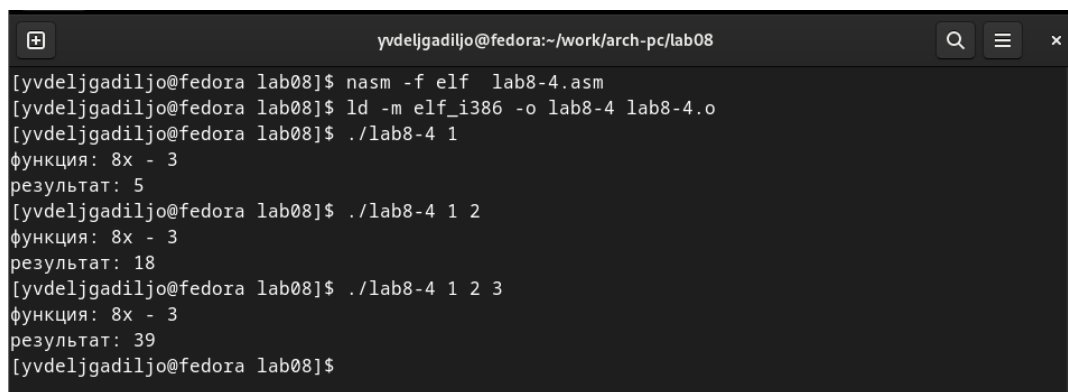
Напишите программу, которая находит сумму значений функции  $f(X)$ . Значения  $x_i$  передаются как аргументы. Создайте исполняемый файл и проверьте его работу на нескольких наборах.



The screenshot shows a gedit window titled "lab8-4.asm" with the following assembly code:

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

Рис. 4.1:



The screenshot shows a terminal window with the following commands and output:

```
[yvdeljgatiljo@fedora lab08]$ nasm -f elf lab8-4.asm
[yvdeljgatiljo@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[yvdeljgatiljo@fedora lab08]$ ./lab8-4 1
функция: 8x - 3
результат: 5
[yvdeljgatiljo@fedora lab08]$ ./lab8-4 1 2
функция: 8x - 3
результат: 18
[yvdeljgatiljo@fedora lab08]$ ./lab8-4 1 2 3
функция: 8x - 3
результат: 39
[yvdeljgatiljo@fedora lab08]$
```

Рис. 4.2:

## 5 Выводы

Были получены по организации циклов и работе со стеком на языке NASM.

## 6 Список литературы

- GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
- GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
- Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
- NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
- Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
- Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
- The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
- Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
- Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
- Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
- Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
- Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.

- Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
- 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](http://www.stolyarov.info/books/asm_unix).
- Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
- Таненбаум Э., Бос Х. Современные операционн