Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Солдатов Алексей

Содержание

# 1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Задание

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

# 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. [1](#tbl:std-dir) приведено краткое описание стандартных каталогов Unix.

Table 1: Описание некоторых каталогов файловой системы GNU Linux

| Имя каталога | Описание каталога |
| --- | --- |
| / | Корневая директория, содержащая всю файловую |
| /bin | Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям |
| /etc | Общесистемные конфигурационные файлы и файлы конфигурации установленных программ |
| /home | Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя |
| /media | Точки монтирования для сменных носителей |
| /root | Домашняя директория пользователя root |
| /tmp | Временные файлы |
| /usr | Вторичная иерархия для данных пользователя |

Более подробно об Unix см. в [1–6].

# 4 Выполнение лабораторной работы

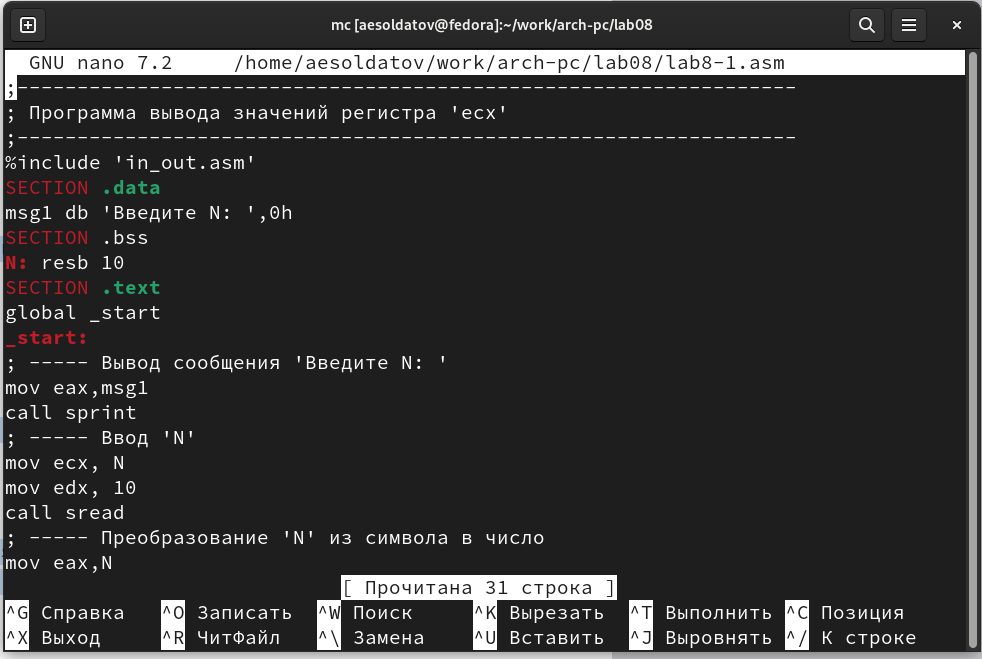
## 4.1 Реализация циклов в NASM

Создал каталог для программ лабораторной работы №8, перешел в него и создал файл “lab8-1.asm” (рис. ??).



Подготовка к работе

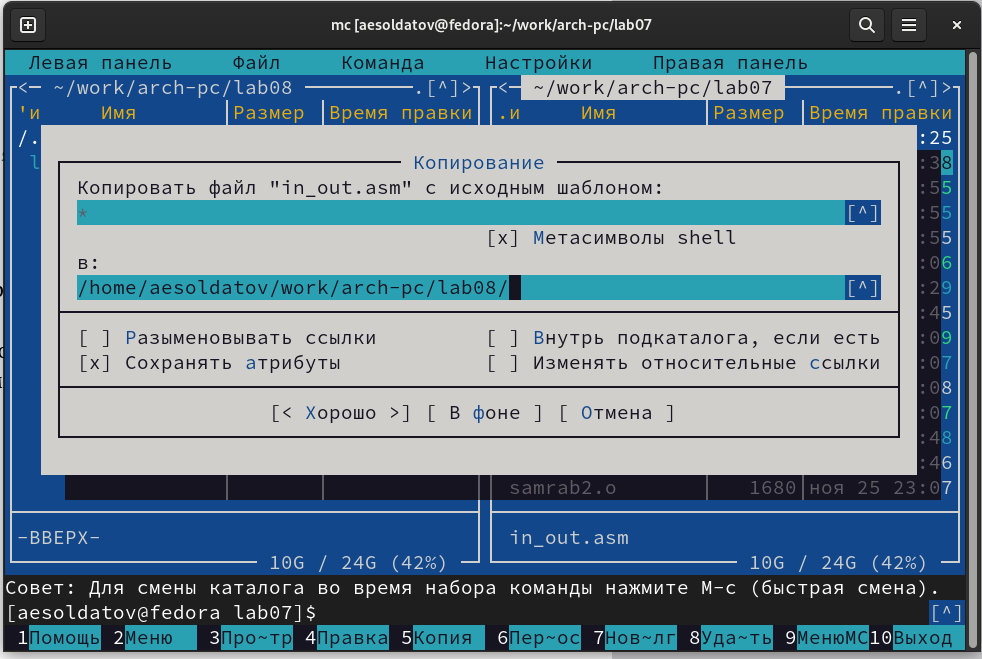
Внимательно изучил текст программы из листинга 8.1 со страницы в ТУИС и ввел в файл “lab8-1.asm” текст программы (рис. ??).



Ввод команд

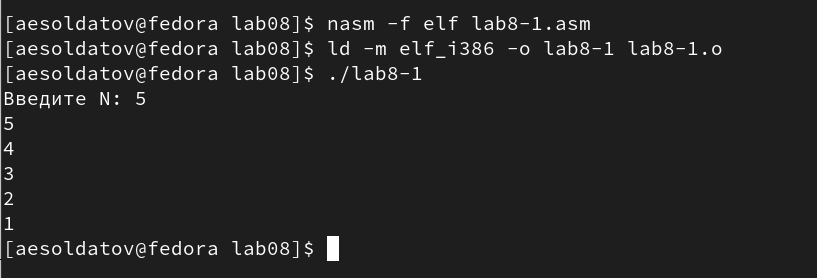
;-----------------------------------------------------------------  
; Программа вывода значений регистра 'ecx'  
;-----------------------------------------------------------------  
%include 'in\_out.asm'  
SECTION .data  
msg1 db 'Введите N: ',0h  
SECTION .bss  
N: resb 10  
SECTION .text  
global \_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:  
mov [N],ecx  
mov eax,[N]  
call iprintLF ; Вывод значения `N`  
loop label ; `ecx=ecx-1` и если `ecx` не '0'  
; переход на `label`  
call quit

Перенес файл “in\_out.asm” из прошлой папки с лабораторной работой в нынешнюю (рис. ??).



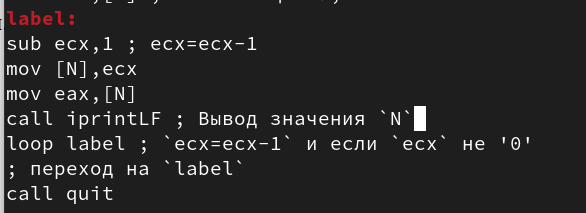
Перенос файла

Создал исполняемый файл и запустил его (рис. ??).



Создание и запуск файла

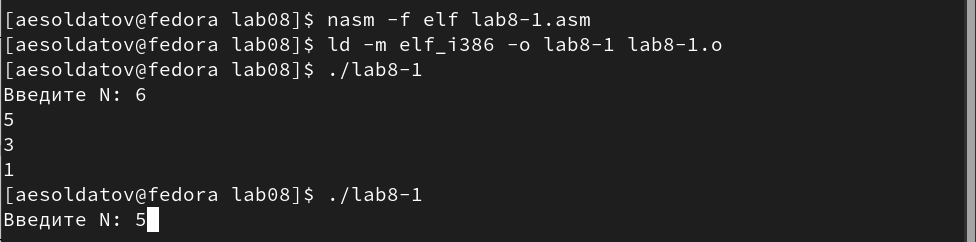
Далее изменил текст программы добавив изменение значение регистра “ecx” в цикле (рис. ??).



Изменение текста программы

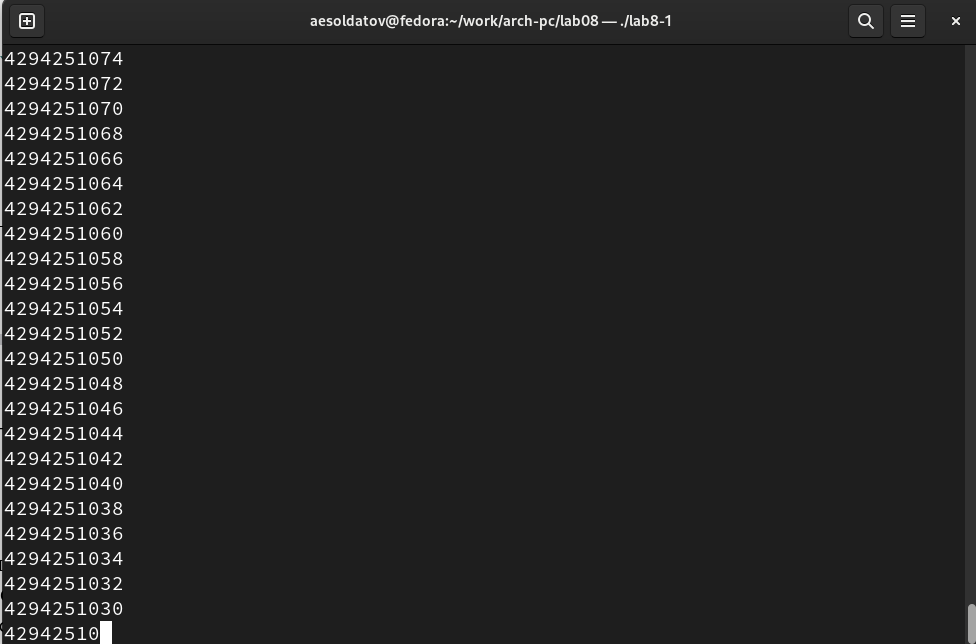
;-----------------------------------------------------------------  
; Программа вывода значений регистра 'ecx'  
;-----------------------------------------------------------------  
%include 'in\_out.asm'  
SECTION .data  
msg1 db 'Введите N: ',0h  
SECTION .bss  
N: resb 10  
SECTION .text  
global \_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 ; Вывод значения `N`  
loop label ; `ecx=ecx-1` и если `ecx` не '0'  
; переход на `label`  
call quit

Создал исполняемый файл и запустил его, программа работает неправильно (рис. ??).



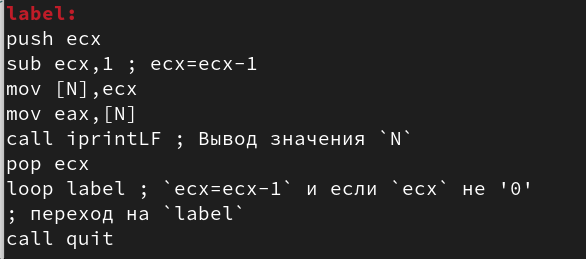
Создание файла

При четном значении N программа выводит значение через один, а при нечетном цикл стал бесконечно выводить непонятные значения, из-за изменения регистра “ecx” в программе (рис. ??).



Бесконечный цикл

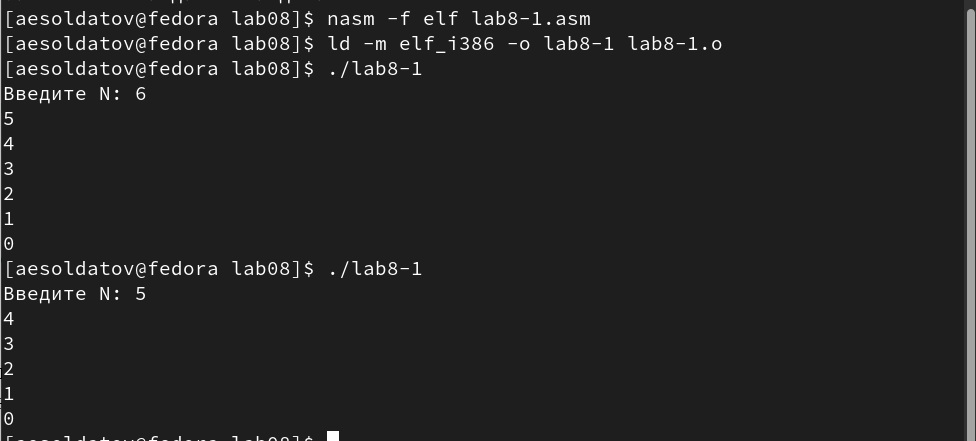
Внес изменения в текст программы добавив команды push и pop (рис. ??).



Изменение текста программы

;-----------------------------------------------------------------  
; Программа вывода значений регистра 'ecx'  
;-----------------------------------------------------------------  
%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:  
push ecx  
sub ecx,1 ; ecx=ecx-1  
mov [N],ecx  
mov eax,[N]  
call iprintLF ; Вывод значения `N`  
pop ecx  
loop label ; `ecx=ecx-1` и если `ecx` не '0'  
; переход на `label`  
call quit

Создал исполняемый файл и запустил его, в данном случае число проходов соответствует значению N, введенному с клавиатуры (рис. ??).



Создание и проверка работы файла

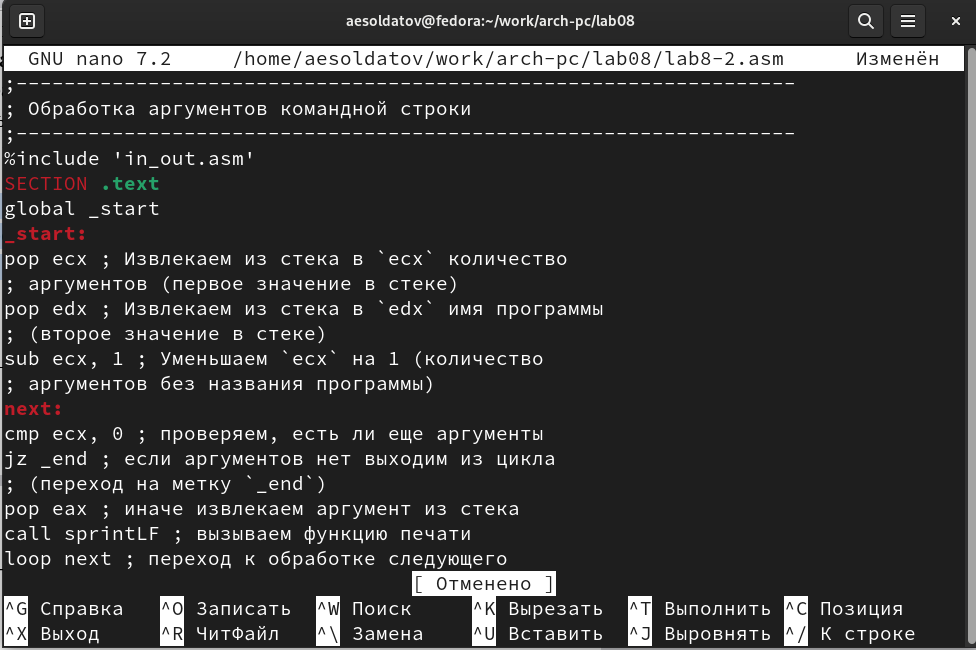
## 4.2 Обработка аргументов командной строки

Создал файл “lab8-2.asm” в каталоге “~/work/arch-pc/lab08” (рис. ??).

Создание файла

Создание файла

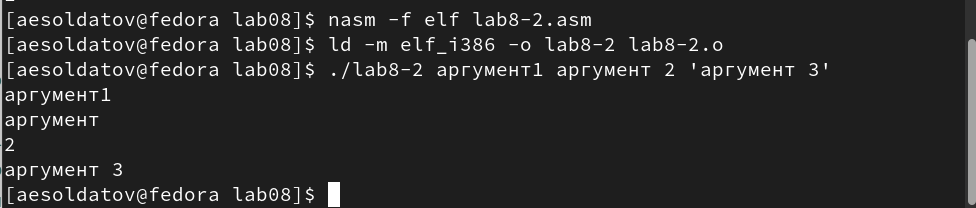
Внимательно изучил текст программы из листинга 8.2 и ввел его в файл (рис. ??).



Ввод команд

;-----------------------------------------------------------------  
; Обработка аргументов командной строки  
;-----------------------------------------------------------------  
%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

Создал исполняемый файл и проверил его работу указав аргументы из примера. Программа обработала 4 аргумента (рис. ??).



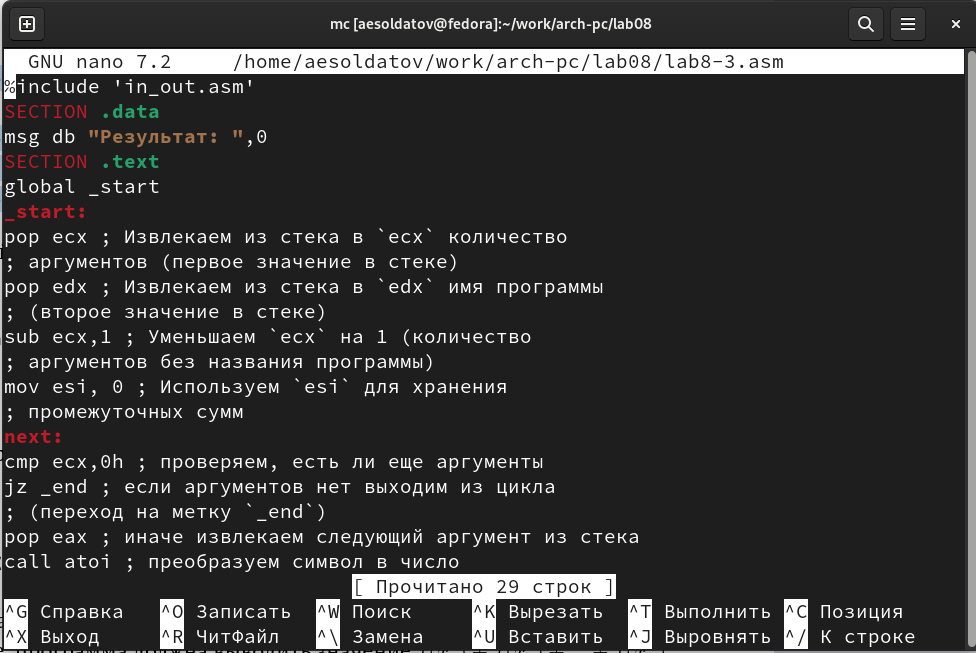
Создание файла

Создал файл “lab8-3.asm” в каталоге “~/work/arch-pc/lab08” (рис. ??).

Создание нового файла

Создание нового файла

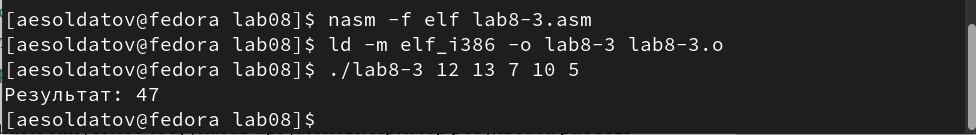
Ввел в него текст программы из листинга 8.3 со страницы в ТУИС (рис. ??).



Ввод команд

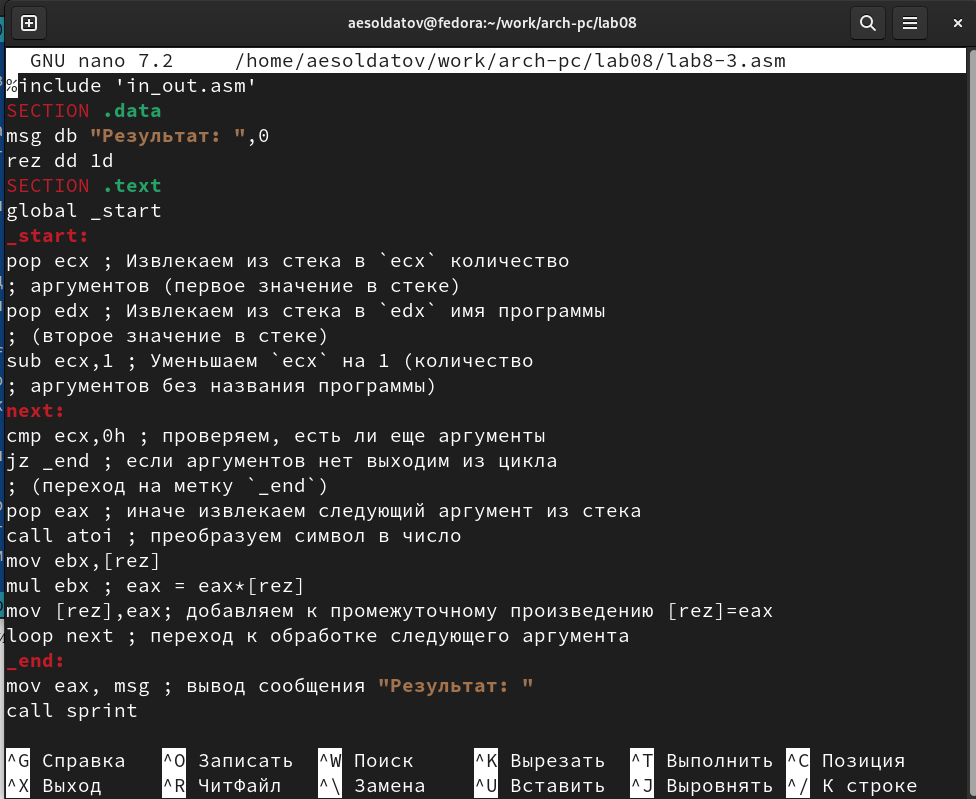
%include 'in\_out.asm'  
SECTION .data  
msg db "Результат: ",0  
SECTION .text  
global \_start  
\_start:  
pop ecx ; Извлекаем из стека в `ecx` количество  
; аргументов (первое значение в стеке)  
pop edx ; Извлекаем из стека в `edx` имя программы  
; (второе значение в стеке)  
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество  
; аргументов без названия программы)  
mov esi, 0 ; Используем `esi` для хранения  
; промежуточных сумм  
next:  
cmp ecx,0h ; проверяем, есть ли еще аргументы  
jz \_end ; если аргументов нет выходим из цикла  
; (переход на метку `\_end`)  
pop eax ; иначе извлекаем следующий аргумент из стека  
call atoi ; преобразуем символ в число  
add esi,eax ; добавляем к промежуточной сумме  
; след. аргумент `esi=esi+eax`  
loop next ; переход к обработке следующего аргумента  
\_end:  
mov eax, msg ; вывод сообщения "Результат: "  
call sprint  
mov eax, esi ; записываем сумму в регистр `eax`  
call iprintLF ; печать результата  
call quit ; завершение программ

Создал исполняемый файл и запустил его, указав аргументы из примера. Программа работает корректно (рис. ??).



Создание и запуск файла

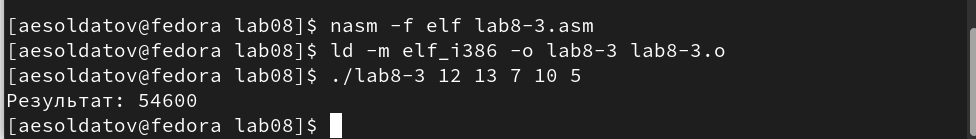
Изменил текст программы для вычисления произведения аргументов (рис. ??).



Изменение текста программы

%include 'in\_out.asm'  
SECTION .data  
msg db "Результат: ",0  
rez dd 1d  
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,0h ; проверяем, есть ли еще аргументы  
jz \_end ; если аргументов нет выходим из цикла  
; (переход на метку `\_end`)  
pop eax ; иначе извлекаем следующий аргумент из стека  
call atoi ; преобразуем символ в число  
mov ebx,[rez]  
mul ebx ; eax = eax\*[rez]  
mov [rez],eax; добавляем к промежуточному произведению [rez]=eax  
loop next ; переход к обработке следующего аргумента  
\_end:  
mov eax, msg ; вывод сообщения "Результат: "  
call sprint  
mov eax, [rez] ; записываем произведение в регистр `eax`  
call iprintLF ; печать результата  
call quit ; завершение программы

Создал и проверил его работу (рис. ??).



Создание файла

## 4.3 Выполнение заданий для самостоятельной работы

Создал файл “samrab.asm” (рис. ??).

Создание файла

Создание файла

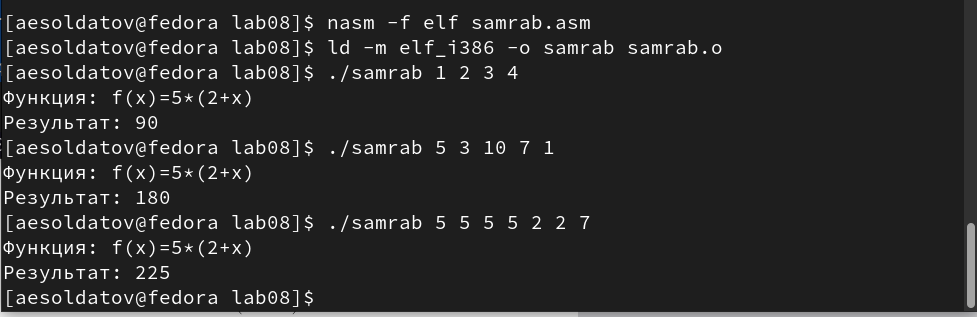
Написал программу нахождения суммы значений функций для разных переменных (рис. ??).



Ввод текста программы

%include 'in\_out.asm'  
SECTION .data  
msgf db "Функция: f(x)=5\*(2+x)",0  
msgr db "Результат: ",0  
rez dd 0d  
SECTION .bss  
rezx resb 80  
SECTION .text  
global \_start  
\_start:  
mov eax,msgf  
call sprintLF  
pop ecx  
pop edx  
sub ecx,1  
next:  
cmp ecx,0  
je \_end  
pop eax  
call atoi  
add eax,2  
mov ebx,5  
mul ebx  
add eax,[rez]  
mov [rez],eax  
loop next  
\_end:  
mov eax,msgr  
call sprint  
mov eax,[rez]  
call iprintLF  
call quit

Создал исполняемый файл и проверил его работу на нескольких наборах переменных (рис. [??]).



Проверка работы программы

# 5 Выводы

Приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.

# Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.

2. Newham C. [Learning the bash Shell: Unix Shell Programming](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658). O’Reilly Media, 2005. 354 с.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Robbins A. [Bash Pocket Reference](https://www.ncbi.nlm.nih.gov/pubmed/25246403). O’Reilly Media, 2016. 156 с.

5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.

6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.