

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Солдатов Алексей

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	15
4.3	Выполнение заданий для самостоятельной работы	21
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1	Подготовка к работе	8
4.2	Ввод команд	9
4.3	Перенос файла	10
4.4	Создание и запуск файла	11
4.5	Изменение текста программы	11
4.6	Создание файла	13
4.7	Бесконечный цикл	13
4.8	Изменение текста программы	14
4.9	Создание и проверка работы файла	15
4.10	Создание файла	16
4.11	Ввод команд	16
4.12	Создание файла	17
4.13	Создание нового файла	17
4.14	Ввод команд	18
4.15	Создание и запуск файла	19
4.16	Изменение текста программы	20
4.17	Создание файла	21
4.18	Создание файла	21
4.19	Ввод текста программы	22
4.20	Проверка работы программы	23

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	7
-----	---	---

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. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

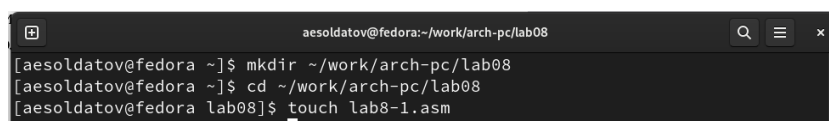
Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит 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

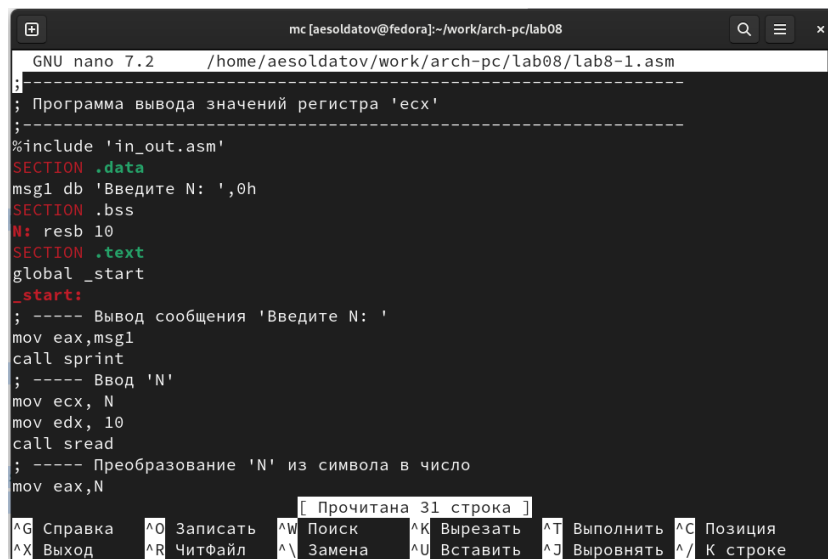
Создал каталог для программ лабораторной работы №8, перешел в него и создал файл “lab8-1.asm” (рис. 4.1).



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

Рис. 4.1: 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. 4.2).



```
mc [aesoldatov@fedora]~/work/arch-pc/lab08
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab08/lab8-1.asm
;-----
; 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
[ Прочитана 31 строка ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/_ К строке
```

Рис. 4.2: 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. 4.3).

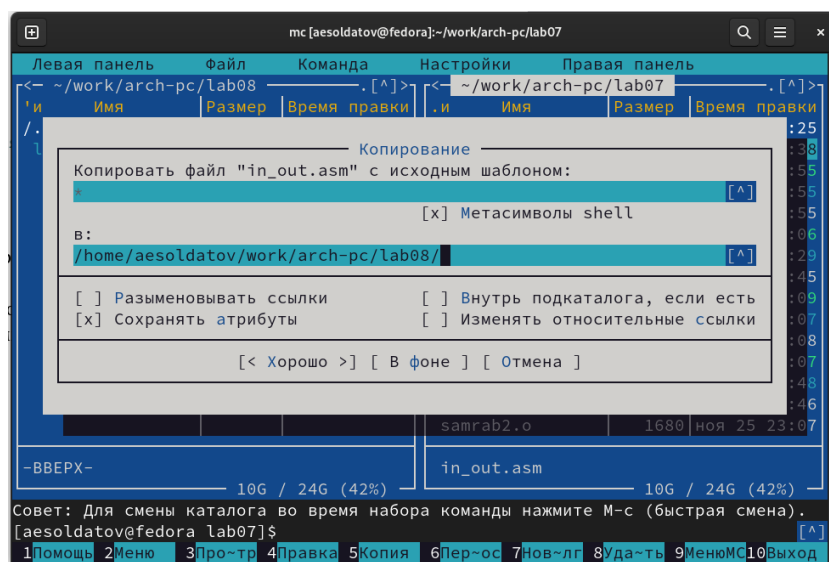


Рис. 4.3: Перенос файла

Создал исполняемый файл и запустил его (рис. 4.4).

```
[aesoldatov@fedora lab08]$ nasm -f elf lab8-1.asm
[aesoldatov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[aesoldatov@fedora lab08]$ ./lab8-1
Введите N: 5
5
4
3
2
1
[aesoldatov@fedora lab08]$
```

Рис. 4.4: Создание и запуск файла

Далее изменил текст программы добавив изменение значение регистра “ecx” в цикле (рис. 4.5).

```
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
```

Рис. 4.5: 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. 4.6).

```

[aesoldatov@fedora lab08]$ nasm -f elf lab8-1.asm
[aesoldatov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[aesoldatov@fedora lab08]$ ./lab8-1
Введите N: 6
5
3
1
[aesoldatov@fedora lab08]$ ./lab8-1
Введите N: 5

```

Рис. 4.6: 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. 4.7).

```

aesoldatov@fedora: ~/work/arch-pc/lab08 — ./lab8-1
4294251074
4294251072
4294251070
4294251068
4294251066
4294251064
4294251062
4294251060
4294251058
4294251056
4294251054
4294251052
4294251050
4294251048
4294251046
4294251044
4294251042
4294251040
4294251038
4294251036
4294251034
4294251032
4294251030

```

Рис. 4.7: Бесконечный цикл

Внес изменения в текст программы добавив команды push и pop (рис. 4.8).

```

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

```

Рис. 4.8: 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. 4.9).

```

[aesoldatov@fedora lab08]$ nasm -f elf lab8-1.asm
[aesoldatov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[aesoldatov@fedora lab08]$ ./lab8-1
Введите N: 6
5
4
3
2
1
0
[aesoldatov@fedora lab08]$ ./lab8-1
Введите N: 5
4
3
2
1
0

```

Рис. 4.9: Создание и проверка работы файла

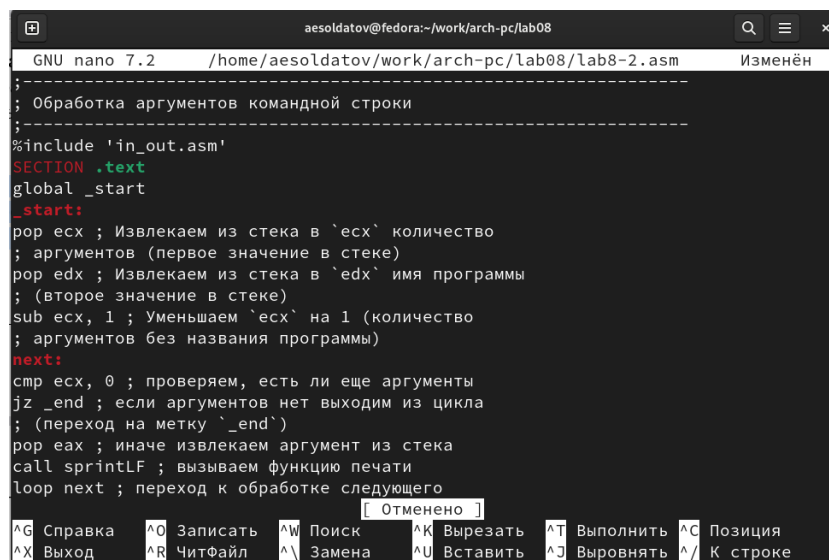
4.2 Обработка аргументов командной строки

Создал файл “lab8-2.asm” в каталоге “~/work/arch-pc/lab08” (рис. 4.10).

```
[aesoldatov@fedora lab08]$ touch lab8-2.asm
```

Рис. 4.10: Создание файла

Внимательно изучил текст программы из листинга 8.2 и ввел его в файл (рис. 4.11).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab08/lab8-2.asm
;-----
; 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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
[ Отменено ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке
```

Рис. 4.11: 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)
```

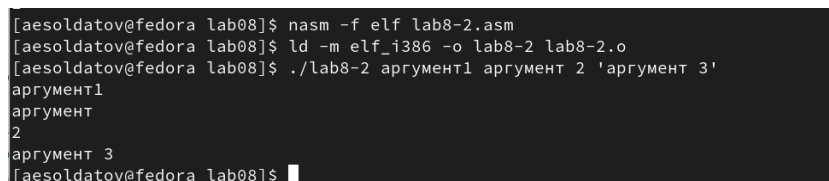


```

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. 4.12).



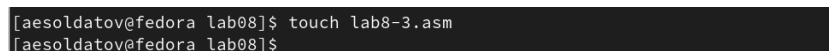
```

[aesoldatov@fedora lab08]$ nasm -f elf lab8-2.asm
[aesoldatov@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[aesoldatov@fedora lab08]$ ./lab8-2 аргумент1 аргумент2 'аргумент 3'
аргумент1
аргумент2
аргумент 3
[aesoldatov@fedora lab08]$

```

Рис. 4.12: Создание файла

Создал файл “lab8-3.asm” в каталоге “~/work/arch-pc/lab08” (рис. 4.13).



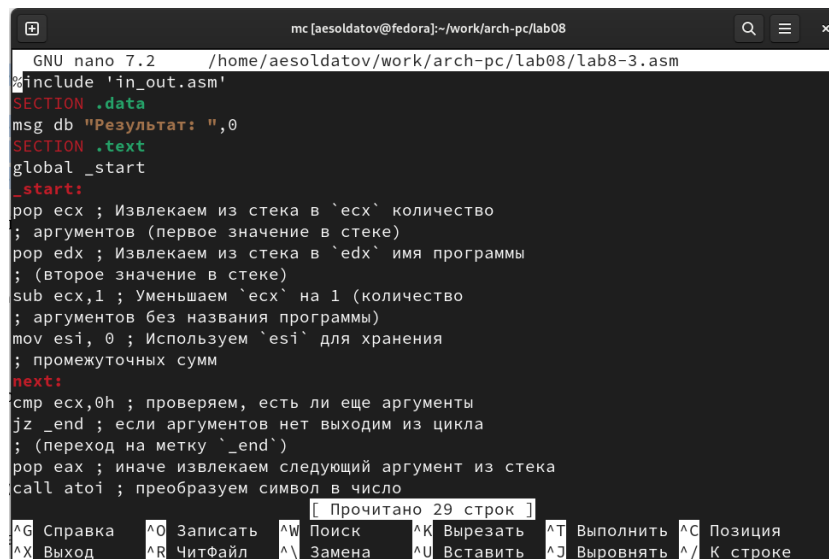
```

[aesoldatov@fedora lab08]$ touch lab8-3.asm
[aesoldatov@fedora lab08]$

```

Рис. 4.13: Создание нового файла

Ввел в него текст программы из листинга 8.3 со страницы в ТУИС (рис. 4.14).



```
mc [aesoldatov@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab08/lab8-3.asm
%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
[ Прочитано 29 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^_ К строке
```

Рис. 4.14: 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. 4.15).

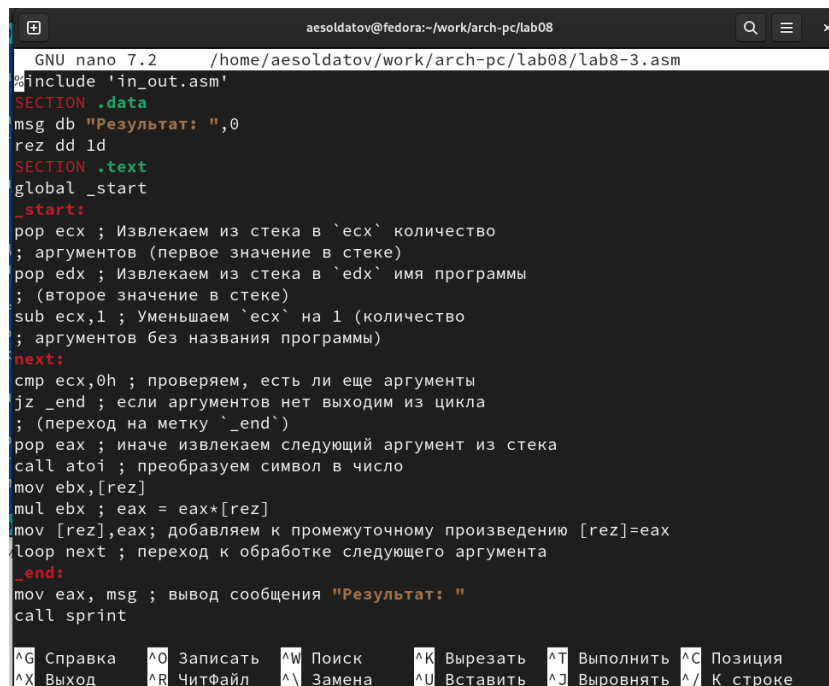
```

[aesoldatov@fedora lab08]$ nasm -f elf lab8-3.asm
[aesoldatov@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[aesoldatov@fedora lab08]$ ./lab8-3 12 13 7 10 5
Результат: 47
[aesoldatov@fedora lab08]$

```

Рис. 4.15: Создание и запуск файла

Изменил текст программы для вычисления произведения аргументов (рис. 4.16).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab08/lab8-3.asm
#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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 4.16: 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. 4.17).

```

[aesoldatov@fedora lab08]$ nasm -f elf lab8-3.asm
[aesoldatov@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[aesoldatov@fedora lab08]$ ./lab8-3 12 13 7 10 5
Результат: 54600
[aesoldatov@fedora lab08]$

```

Рис. 4.17: Создание файла

4.3 Выполнение заданий для самостоятельной работы

Создал файл “samrab.asm” (рис. 4.18).

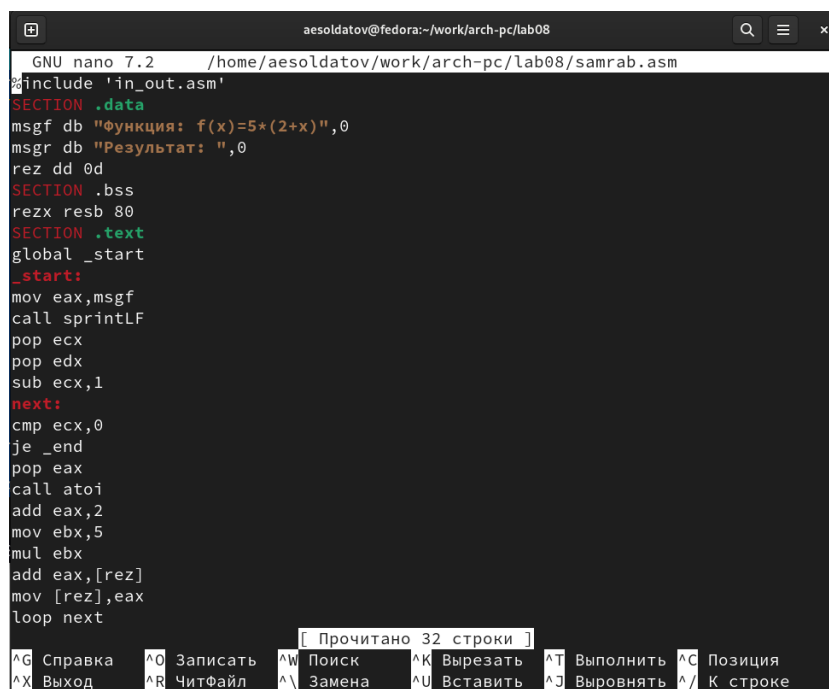
```

[aesoldatov@fedora lab08]$ touch samrab.asm
[aesoldatov@fedora lab08]$

```

Рис. 4.18: Создание файла

Написал программу нахождения суммы значений функций для разных переменных (рис. 4.19).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab08/samrab.asm
#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
```

[Прочитано 32 строки]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/ К строке

Рис. 4.19: 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,msgnr
call sprint
mov eax,[rez]
call iprintLF
call quit

```

Создал исполняемый файл и проверил его работу на нескольких наборах переменных (рис. [4.20]).

```

[aesoldatov@fedora lab08]$ nasm -f elf samrab.asm
[aesoldatov@fedora lab08]$ ld -m elf_i386 -o samrab samrab.o
[aesoldatov@fedora lab08]$ ./samrab 1 2 3 4
Функция:  $f(x)=5*(2+x)$ 
Результат: 90
[aesoldatov@fedora lab08]$ ./samrab 5 3 10 7 1
Функция:  $f(x)=5*(2+x)$ 
Результат: 180
[aesoldatov@fedora lab08]$ ./samrab 5 5 5 5 2 2 7
Функция:  $f(x)=5*(2+x)$ 
Результат: 225
[aesoldatov@fedora lab08]$

```

Рис. 4.20: 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. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. 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.