

Отчёт по лабораторной работе 8

Архитектура компьютеров

Агджабекова Эся Рустамовна НПИбд-01-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Самостоятельное задание	16
3	Выводы	19

Список иллюстраций

2.1	Программа в файле lab8-1.asm	7
2.2	Запуск программы lab8-1.asm	8
2.3	Программа в файле lab8-1.asm	9
2.4	Запуск программы lab8-1.asm	10
2.5	Программа в файле lab8-1.asm	11
2.6	Запуск программы lab8-1.asm	12
2.7	Программа в файле lab8-2.asm	13
2.8	Запуск программы lab8-2.asm	13
2.9	Программа в файле lab8-3.asm	14
2.10	Запуск программы lab8-3.asm	14
2.11	Программа в файле lab8-3.asm	15
2.12	Запуск программы lab8-3.asm	16
2.13	Программа в файле prog.asm	17
2.14	Запуск программы prog.asm	18

Список таблиц

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

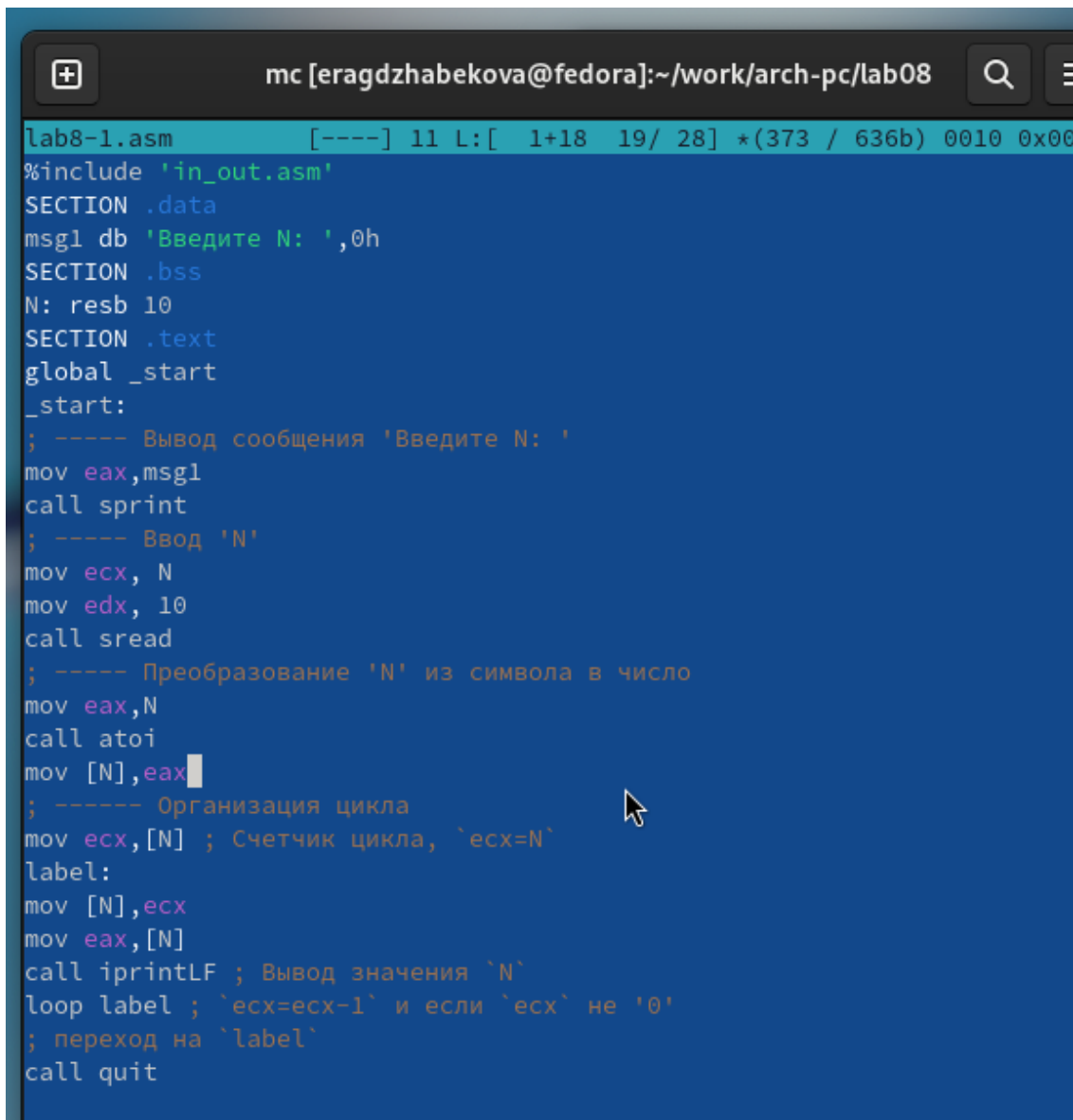
2 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы №8 и файл lab8-1.asm.

При реализации циклов в NASM с использованием инструкции `loop` следует учитывать, что данная инструкция использует регистр `ecx` в качестве счетчика и на каждом шаге уменьшает его значение на единицу. Рассмотрим пример программы, выводящей значение регистра `ecx`.

Добавила в файл lab8-1.asm текст программы из листинга 8.1 (рис. 2.1).

Создала исполняемый файл и проверила его работу (рис. 2.2).



```
lab8-1.asm [----] 11 L: [ 1+18 19/ 28] *(373 / 636b) 0010 0x00
#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
```

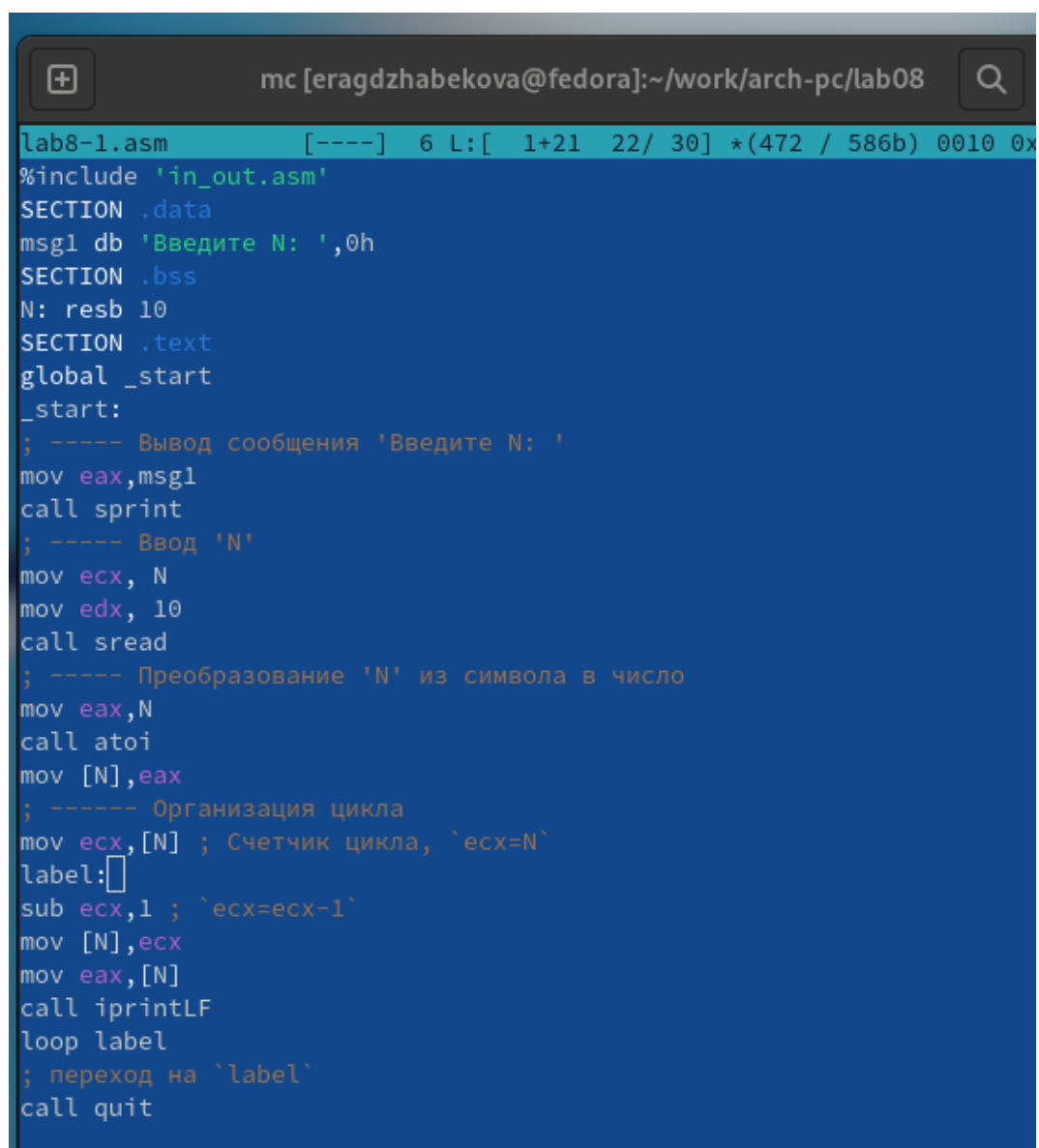
Рис. 2.1: Программа в файле lab8-1.asm

```
eragdzhbekova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
eragdzhbekova@fedora:~/work/arch-pc/lab08$ □
```

Рис. 2.2: Запуск программы lab8-1.asm

Пример демонстрирует, что использование регистра `ecx` внутри цикла `loop` может привести к некорректной работе программы. Изменила текст программы, добавив модификацию значения регистра `ecx` внутри цикла (рис. 2.3).

Теперь программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N` (рис. 2.4).



```
lab8-1.asm [----] 6 L: [ 1+21 22/ 30] *(472 / 586b) 0010 0x
#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
; переход на `label`
call quit
```

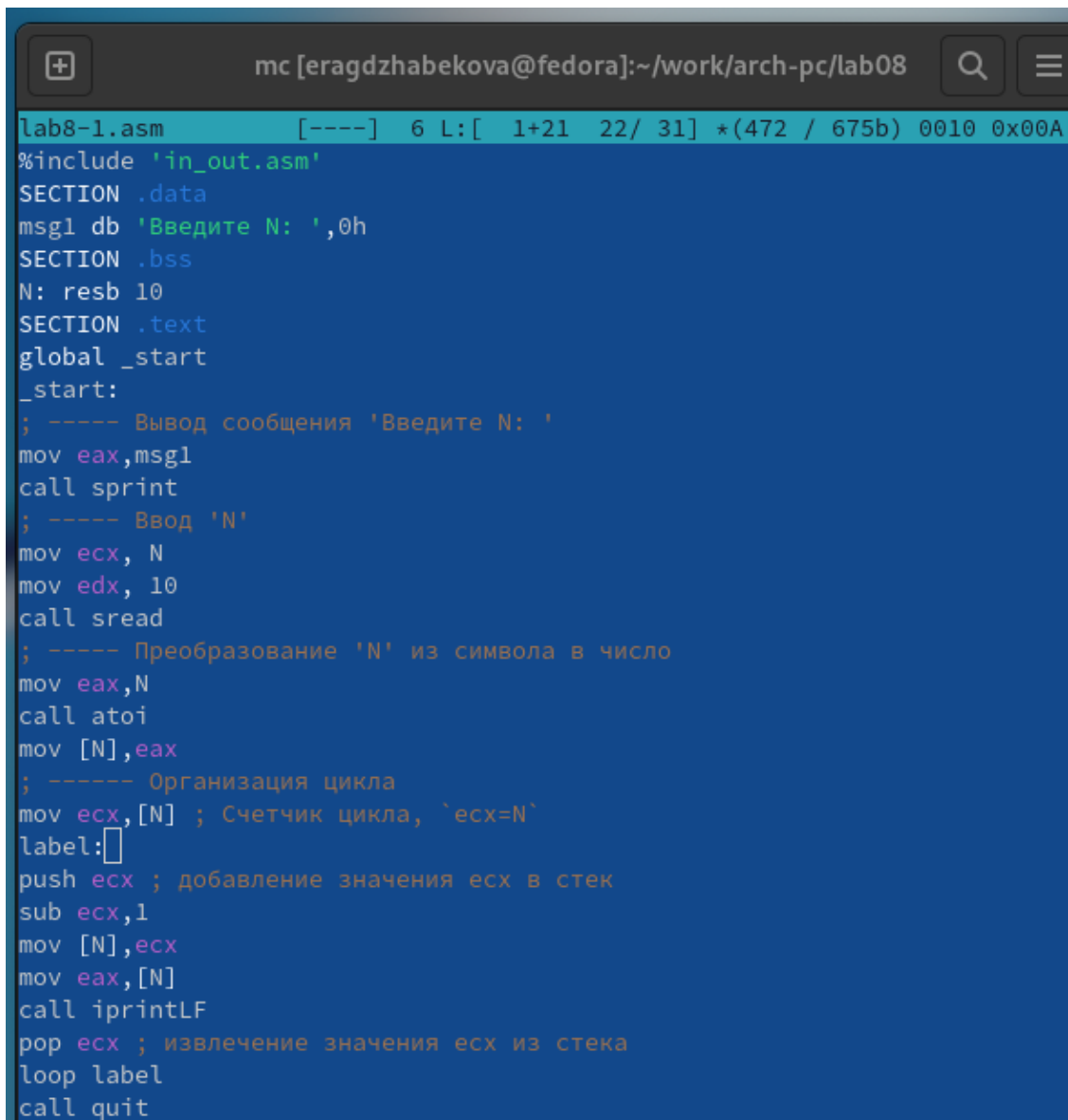
Рис. 2.3: Программа в файле lab8-1.asm

```
4294942274
4294942272
4294942270
4294942268
4294942266
4294942264
4294942262
4294942260
4^C
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
eragdzhbekova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.4: Запуск программы lab8-1.asm

Для корректной работы программы при использовании регистра `ecx` в цикле можно применять стек. Внесла изменения в текст программы, добавив команды `push` и `pop` для сохранения значения регистра `ecx` (рис. 2.5).

Создала исполняемый файл и проверила его работу. Программа корректно выводит числа от $N-1$ до 0, при этом число проходов цикла соответствует значению N (рис. 2.6).



```
lab8-1.asm [-----] 6 L:[ 1+21 22/ 31] *(472 / 675b) 0010 0x00A
#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 ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
call quit
```

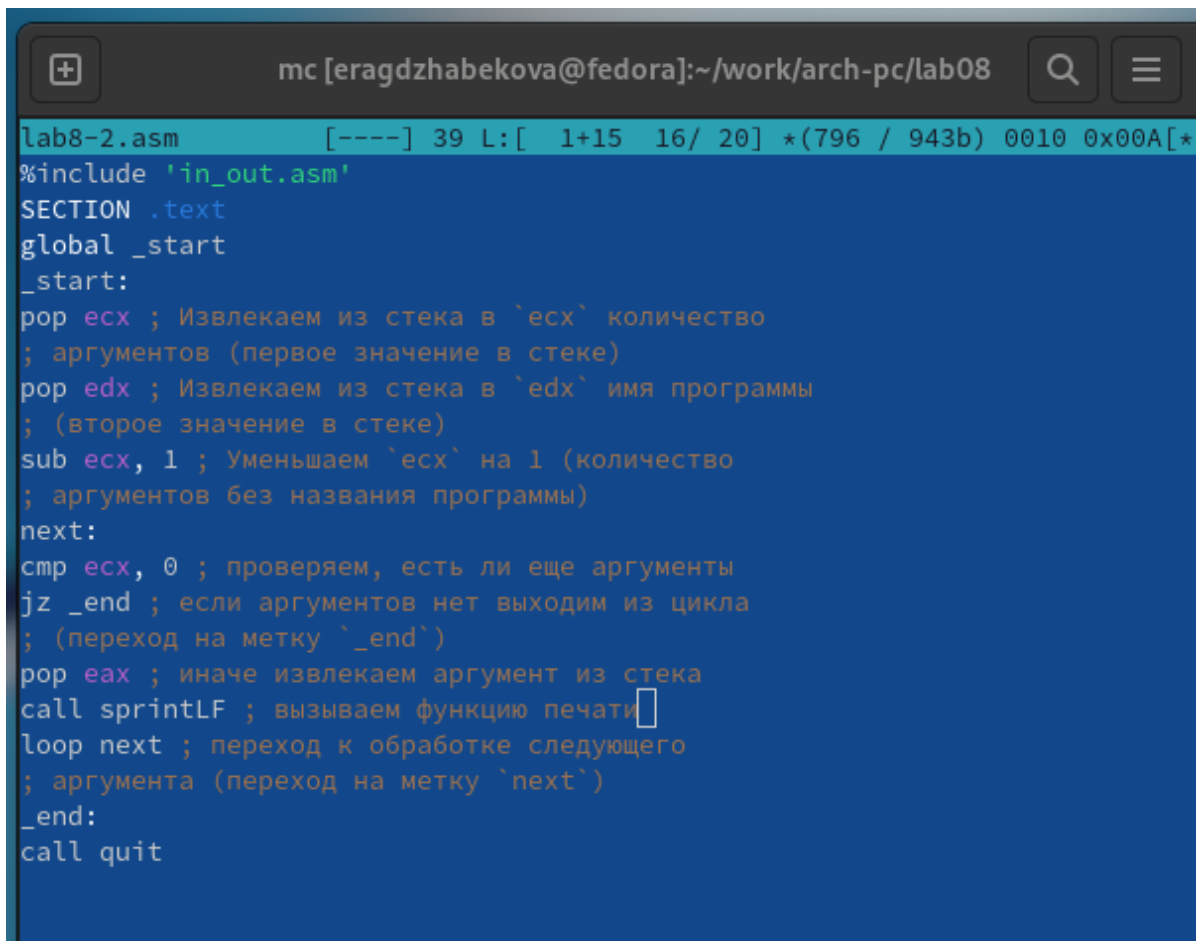
Рис. 2.5: Программа в файле lab8-1.asm

```
eragdzhbekova@fedora:~/work/arch-pc/lab08$  
eragdzhbekova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 4  
3  
2  
1  
0  
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 5  
4  
3  
2  
1  
0  
eragdzhbekova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.6: Запуск программы lab8-1.asm

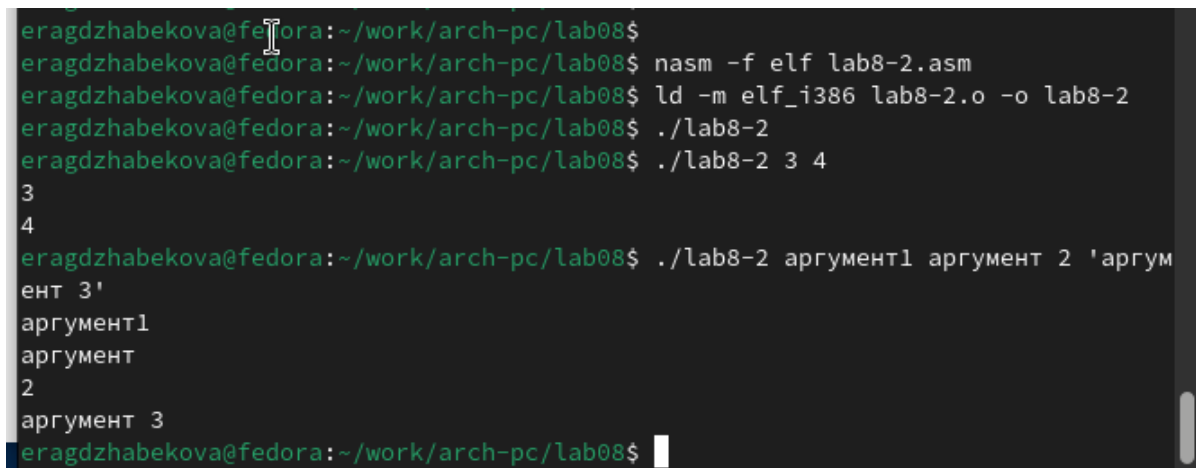
Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.2 (рис. 2.7).

Создала исполняемый файл и запустила его, указав аргументы. Программа обработала 4 аргумента, которые интерпретируются как слова/числа, разделенные пробелом (рис. 2.8).



```
mc [eragdzhbekova@fedora]:~/work/arch-pc/lab08
lab8-2.asm [----] 39 L: [ 1+15 16/ 20] *(796 / 943b) 0010 0x00A[*]
#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. 2.7: Программа в файле lab8-2.asm

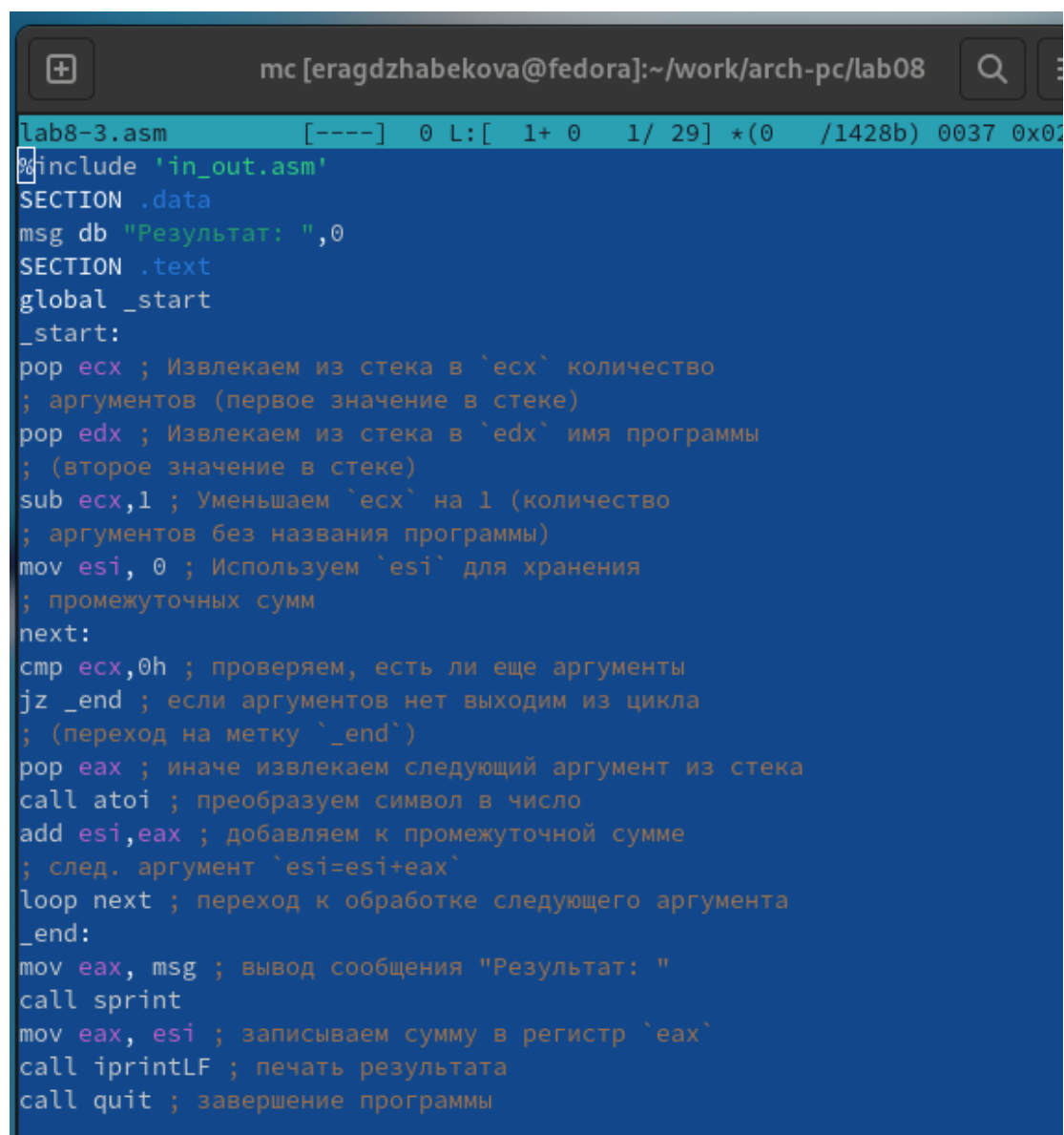


```
eragdzhbekova@fedora:~/work/arch-pc/lab08$
eragdzhbekova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-2
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-2 3 4
3
4
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргум
ент 3'
аргумент1
аргумент
2
аргумент 3
eragdzhbekova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.8: Запуск программы lab8-2.asm

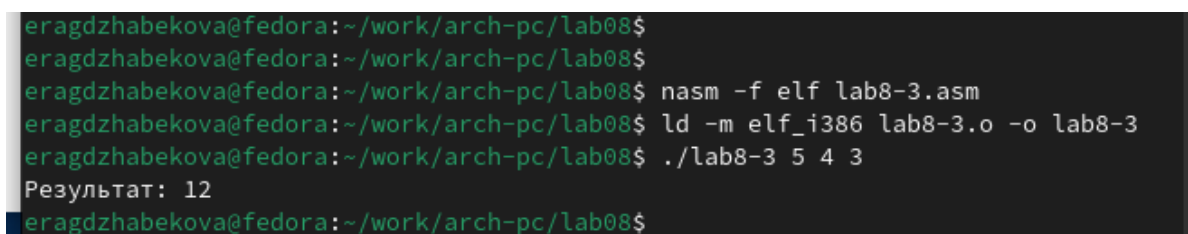
Рассмотрим еще один пример программы, вычисляющей сумму чисел, пере-

данных в качестве аргументов командной строки (рис. 2.9, 2.10).



```
lab8-3.asm [----] 0 L:[ 1+ 0 1/ 29] *(0 /1428b) 0037 0x02
%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 программы
```

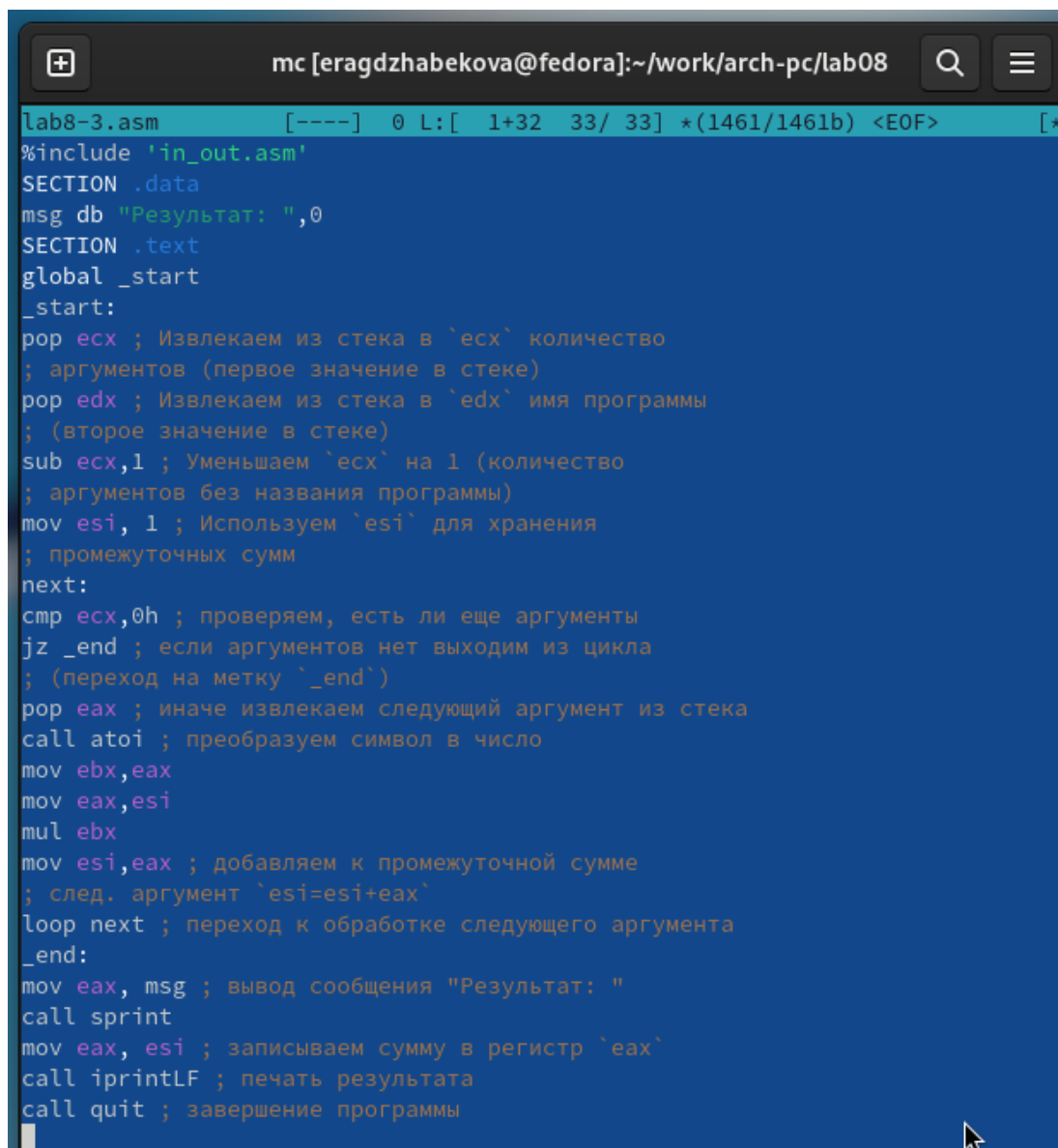
Рис. 2.9: Программа в файле lab8-3.asm



```
eragdzhbekova@fedora:~/work/arch-pc/lab08$
eragdzhbekova@fedora:~/work/arch-pc/lab08$
eragdzhbekova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 4 3
Результат: 12
eragdzhbekova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.10: Запуск программы lab8-3.asm

Изменила текст программы из листинга 8.3, чтобы вычислять произведение аргументов командной строки (рис. 2.11, 2.12).



```
lab8-3.asm [----] 0 L: [ 1+32 33/ 33] *(1461/1461b) <EOF> [*]
#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, 1 ; 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
mov ebx,eax
mov eax,esi
mul ebx
mov 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 программы
```

Рис. 2.11: Программа в файле lab8-3.asm

```

eragdzhbekova@fedora:~/work/arch-pc/lab08$
eragdzhbekova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 4 3
Результат: 12
eragdzhbekova@fedora:~/work/arch-pc/lab08$
eragdzhbekova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 4 3
Результат: 60
eragdzhbekova@fedora:~/work/arch-pc/lab08$

```

Рис. 2.12: Запуск программы lab8-3.asm

2.1 Самостоятельное задание

Разработала программу для вычисления суммы значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$.

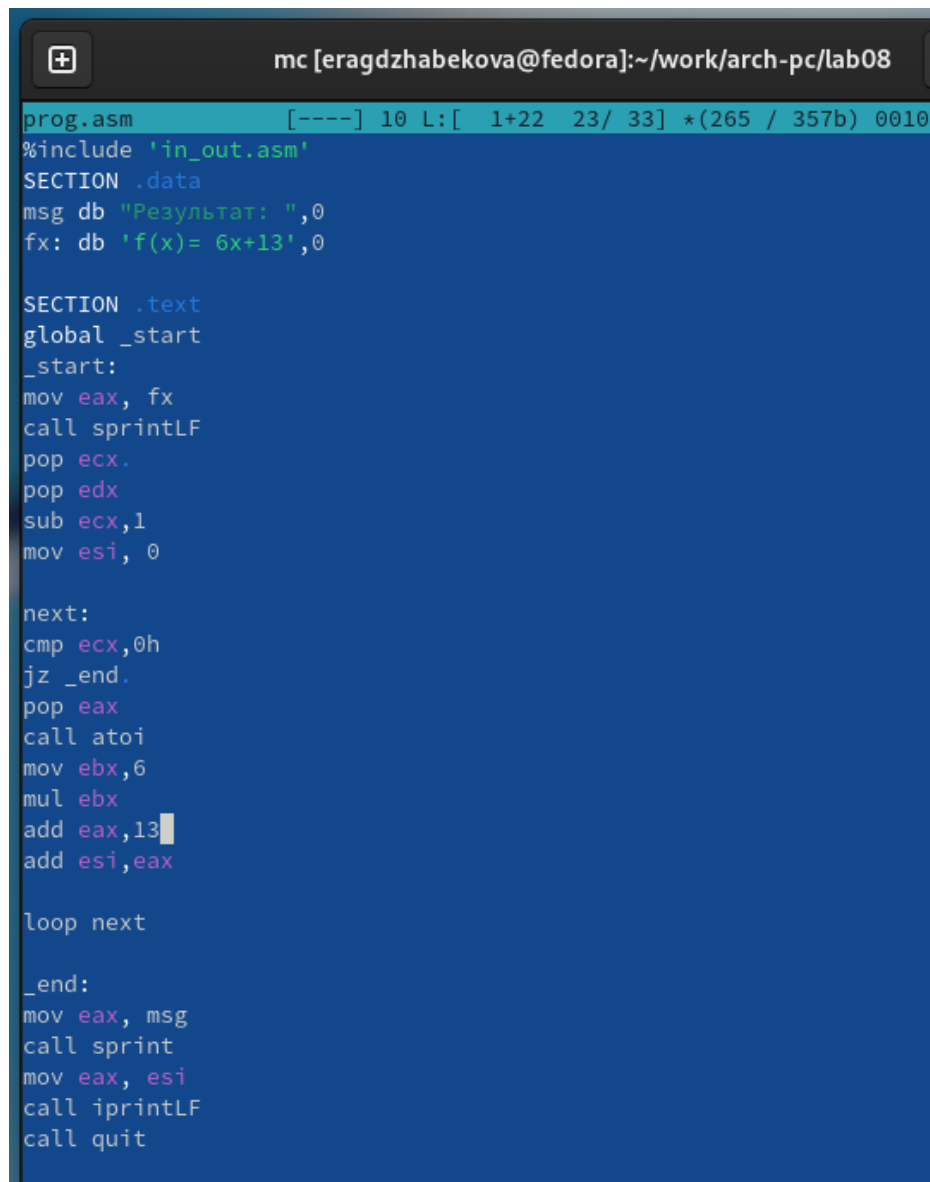
Программа выводит результат вычисления $f(x_1) + f(x_2) + \dots + f(x_n)$, где значения x передаются в качестве аргументов.

Функция $f(x)$ соответствует варианту задания из лабораторной работы №7.

Для варианта 15:

$$f(x) = 6x + 13$$

Создала исполняемый файл и проверила работу программы на различных наборах значений x (рис. 2.13, 2.14).



```
mc [eragdzhabekova@fedora]:~/work/arch-pc/lab08
prog.asm [----] 10 L: [ 1+22 23/ 33] *(265 / 357b) 0010
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 6x+13',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx,6
mul ebx
add eax,13
add esi,eax

loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 2.13: Программа в файле prog.asm

Проверила корректность работы, запустив программу сначала с одним аргументом.

Например, при подстановке $f(0) = 13$, $f(3) = 31$.

Затем протестировала с несколькими аргументами, убедившись в правильности вычисления суммы значений функции.

```

eragdzhbekova@fedora:~/work/arch-pc/lab08$
eragdzhbekova@fedora:~/work/arch-pc/lab08$ nasm -f elf prog.asm
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 prog.o -o prog
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./prog
f(x)= 6x+13
Результат: 0
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./prog 1
f(x)= 6x+13
Результат: 19
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./prog 2
f(x)= 6x+13
Результат: 25
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./prog 0
f(x)= 6x+13
Результат: 13
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./prog 3
f(x)= 6x+13
Результат: 31
eragdzhbekova@fedora:~/work/arch-pc/lab08$ ./prog 1 3 4 1
f(x)= 6x+13
Результат: 106
eragdzhbekova@fedora:~/work/arch-pc/lab08$

```

Рис. 2.14: Запуск программы prog.asm

3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере `nasn`.