

Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютерных наук

Литвинов Максим Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация циклов в NASM	6
2.2	Обработка аргументов командной строки	12
2.3	Задание для самостоятельной работы	17
3	Выводы	20

Список иллюстраций

2.1	Изменение кода	7
2.2	Запуск программы	8
2.3	Изменение кода	9
2.4	Запуск программы	10
2.5	Изменение кода	11
2.6	Запуск программы	12
2.7	Изменение кода	13
2.8	Запуск программы	13
2.9	Изменение кода	14
2.10	Запуск программы	15
2.11	Изменение кода	16
2.12	Запуск программы	17
2.13	Изменение кода	18
2.14	Запуск программы	19

Список таблиц

1 Цель работы

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

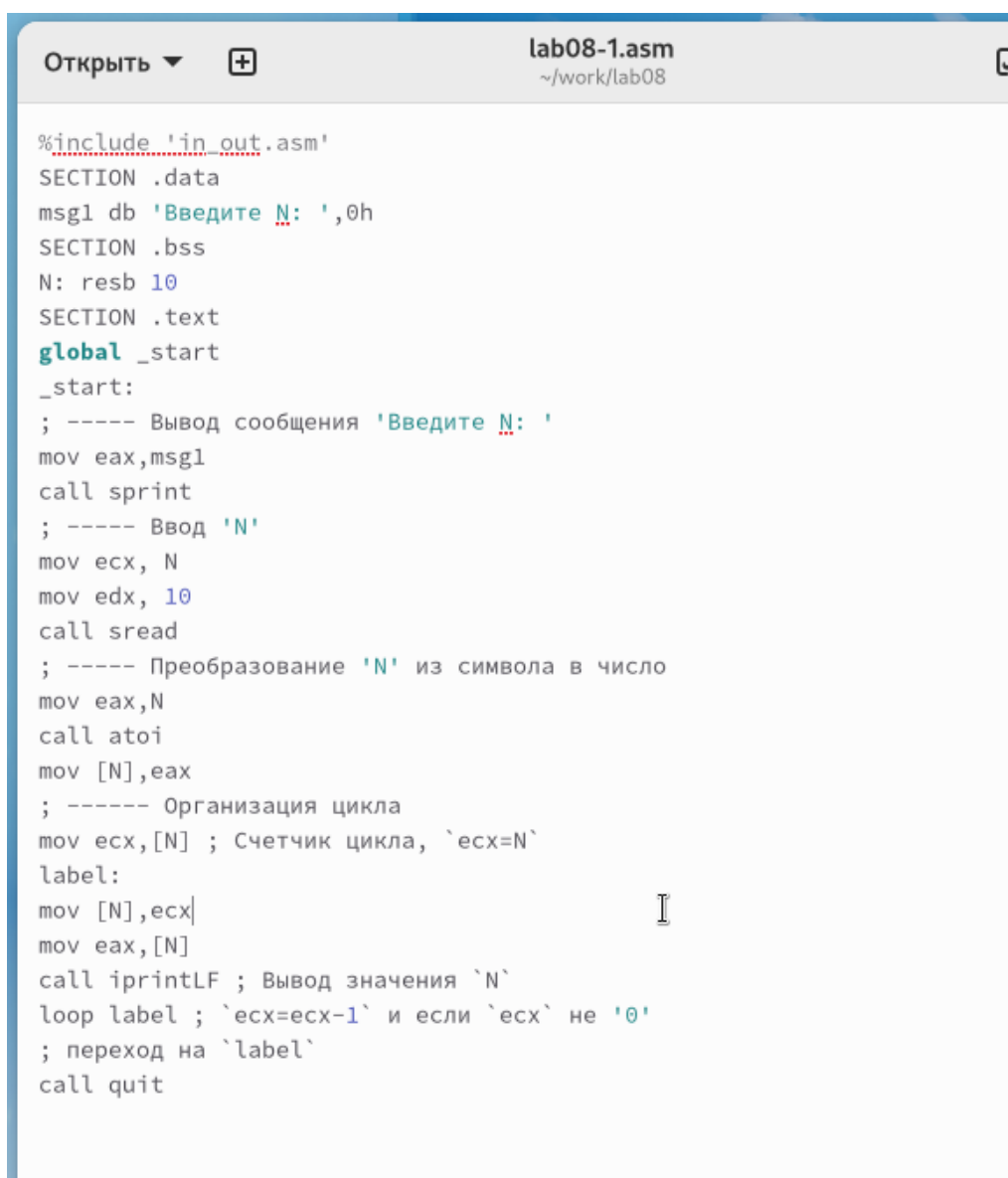
2 Выполнение лабораторной работы

2.1 Реализация циклов в NASM

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

При использовании инструкции `loop` в NASM для реализации циклов, необходимо учитывать, что она использует регистр `ecx` в качестве счетчика и на каждом шаге уменьшает его значение на единицу. Для наглядности рассмотрим программу, которая выводит значение регистра `ecx`.

Написал в файл lab8-1.asm текст программы из листинга 8.1. Создал исполняемый файл и проверил его работу.



```
Открыть ▾ + lab08-1.asm ~/work/lab08


%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: Изменение кода

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

Рис. 2.2: Запуск программы

Однако, в данном примере становится очевидно, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Чтобы исправить это, мы можем использовать стек для сохранения значения счетчика цикла `loop`. Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`.

Открыть ▾ 

lab08-1.asm
~/work/lab08

```
%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
```

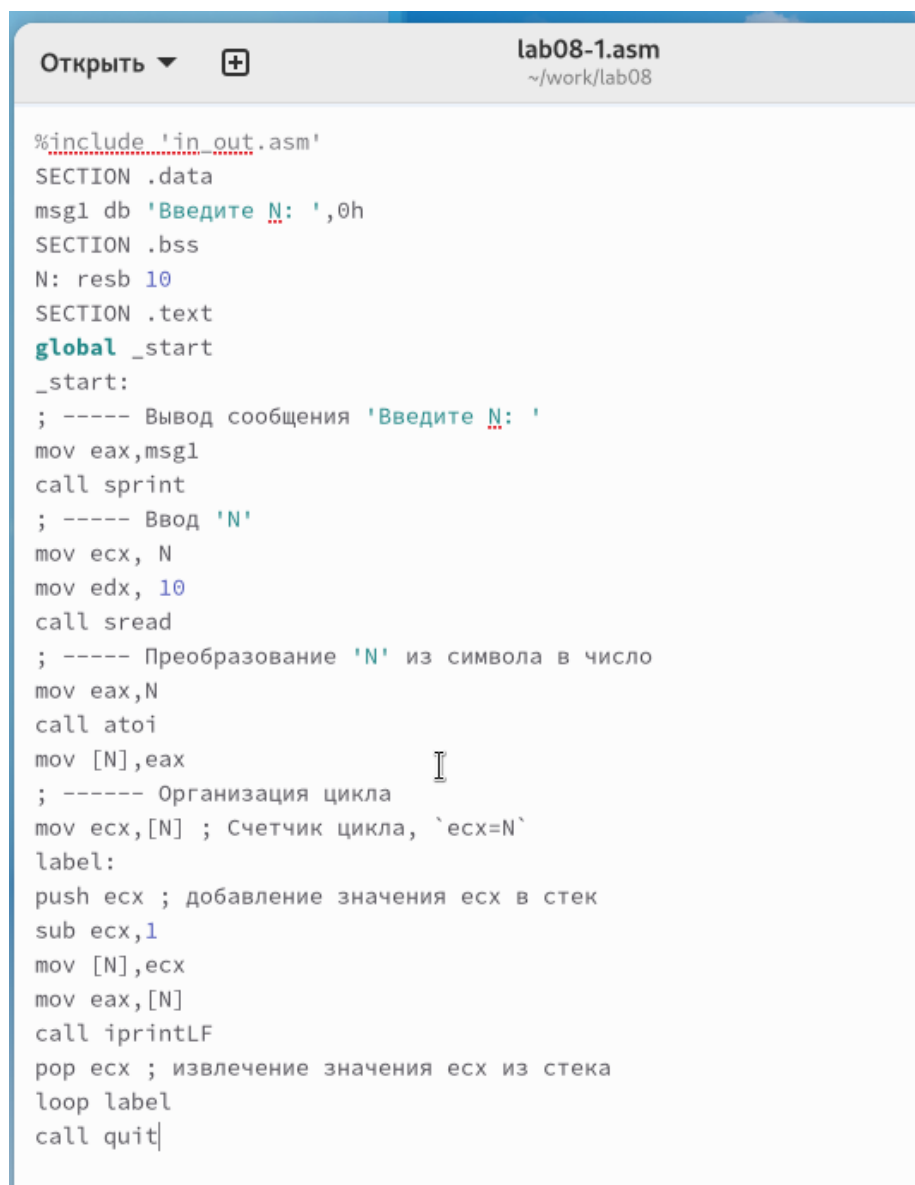
Рис. 2.3: Изменение кода

```
4294929290
4294929288
4294929286
4294929284
4294929282
4294929280
4294929278
42949292^C
[malitvinov@fedora lab08]$ ./lab08-1
Введите N: 4
3
1
[malitvinov@fedora lab08]$
```

Рис. 2.4: Запуск программы

Внесем изменения в текст программы, добавив команды `push` и `pop`, чтобы сохранить и извлечь значение счетчика из стека соответственно. После этого создадим исполняемый файл и проверим его работу. Таким образом, мы обеспечим корректность работы программы.

Создал исполняемый файл и проверьте его работу. Программа выводит числа от $N-1$ до 0, число проходов цикла соответствует N .



```
Открыть ▾ + lab08-1.asm
~/work/lab08

%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: Изменение кода

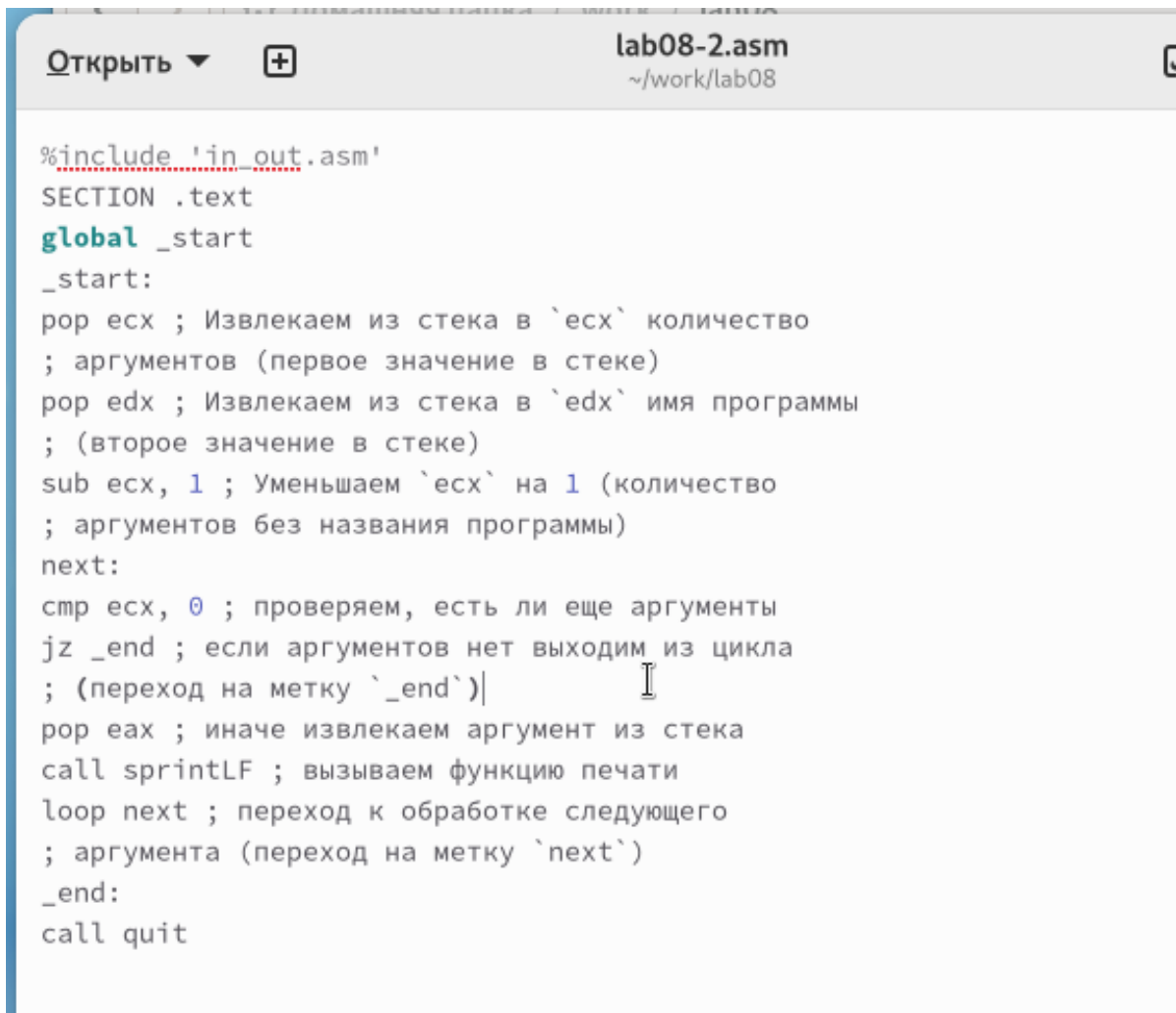
```
[malitvinov@fedora lab08]$  
[malitvinov@fedora lab08]$ nasm -f elf lab08-1.asm  
[malitvinov@fedora lab08]$ ld -m elf_i386 lab08-1.o -o lab08-1  
[malitvinov@fedora lab08]$ ./lab08-1  
Введите N: 4  
3  
2  
1  
0  
[malitvinov@fedora lab08]$
```

Рис. 2.6: Запуск программы

2.2 Обработка аргументов командной строки

Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2.

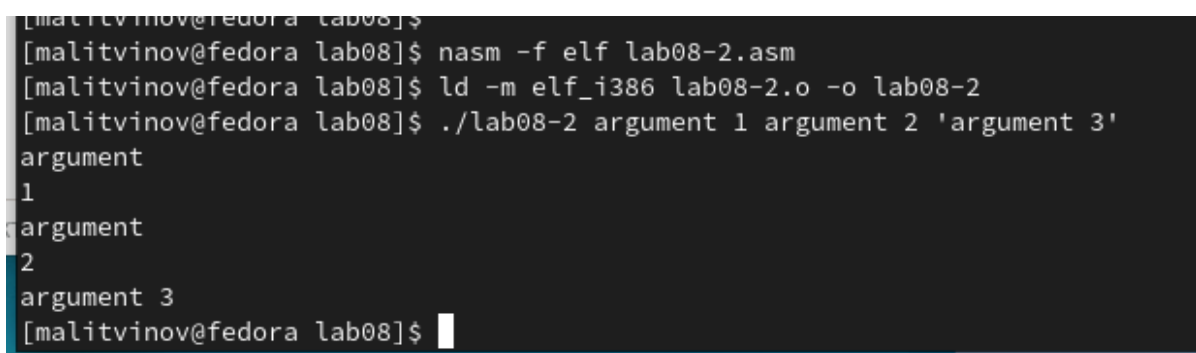
Создал исполняемый файл и запустил его, указав аргументы. Программа обработала 5 аргументов. Аргументами считаются слова/числа, разделенные пробелом.



```
Открыть ▾ + lab08-2.asm
~/work/lab08

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

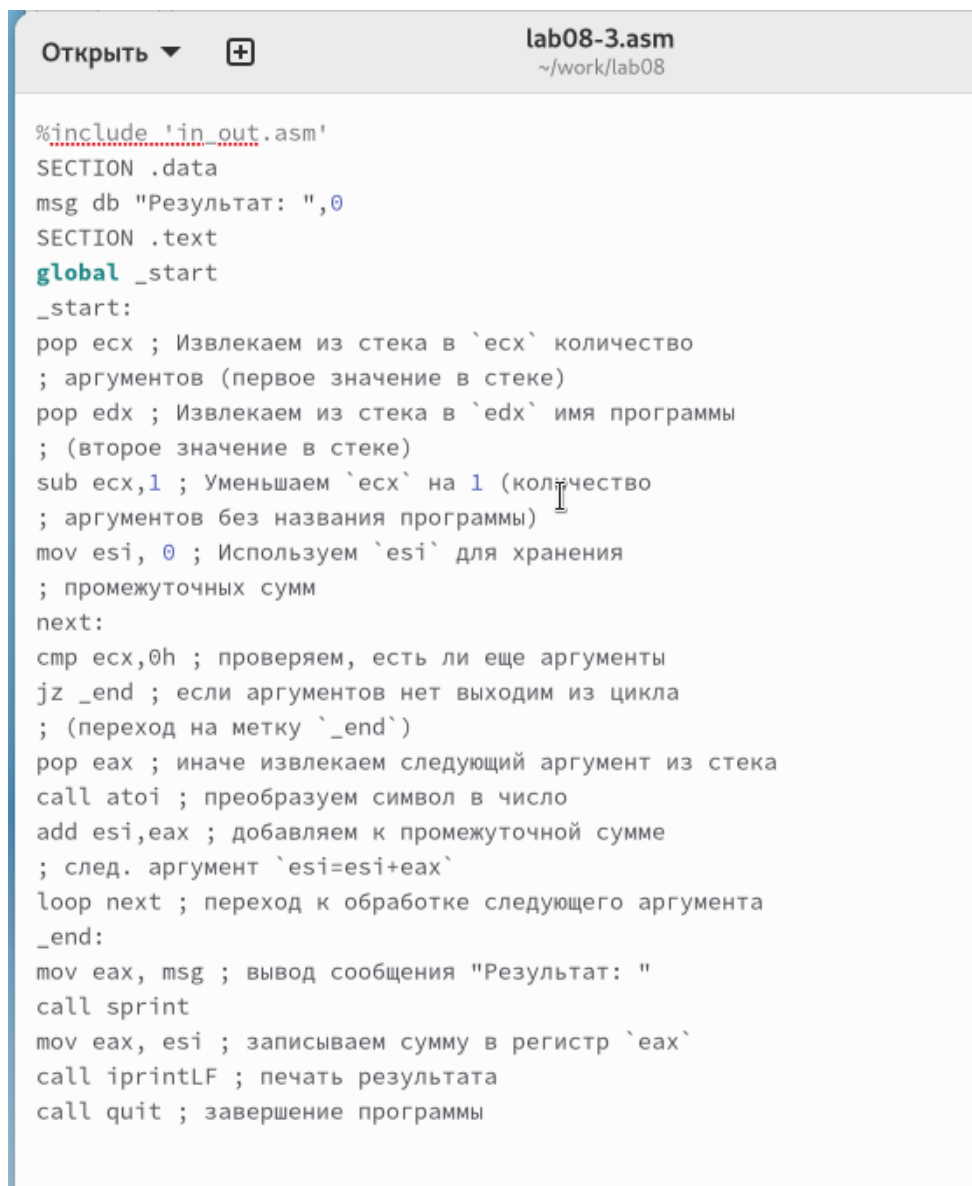
Рис. 2.7: Изменение кода




```
[malitvinov@fedora lab08]$
[malitvinov@fedora lab08]$ nasm -f elf lab08-2.asm
[malitvinov@fedora lab08]$ ld -m elf_i386 lab08-2.o -o lab08-2
[malitvinov@fedora lab08]$ ./lab08-2 argument 1 argument 2 'argument 3'
argument
1
argument
2
argument 3
[malitvinov@fedora lab08]$
```

Рис. 2.8: Запуск программы

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.



```
Открыть ▾  lab08-3.asm
~/.work/lab08

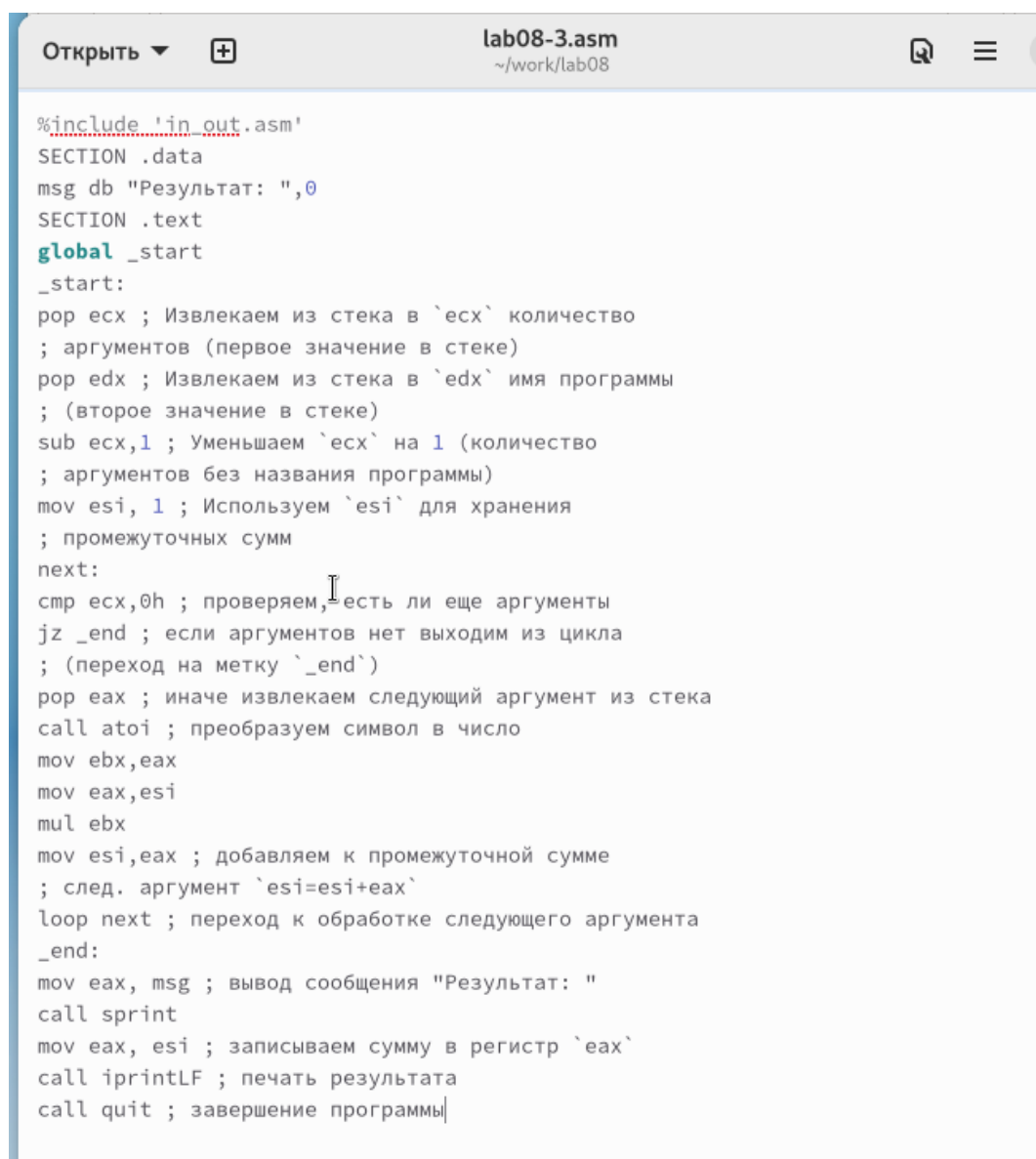
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; 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 iprintf ; печать результата
call quit ; завершение программы
```

Рис. 2.9: Изменение кода

```
[malitvinov@fedora lab08]$ nasm -f elf lab08-3.asm
[malitvinov@fedora lab08]$ ld -m elf_i386 lab08-3.o -o lab08-3
[malitvinov@fedora lab08]$ ./lab08-3
Результат: 0
[malitvinov@fedora lab08]$ ./lab08-3 3 4 5 6
Результат: 18
[malitvinov@fedora lab08]$
```

Рис. 2.10: Запуск программы

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



```
lab08-3.asm
~/.work/lab08

Открыть +

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por 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`)
por 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: Изменение кода

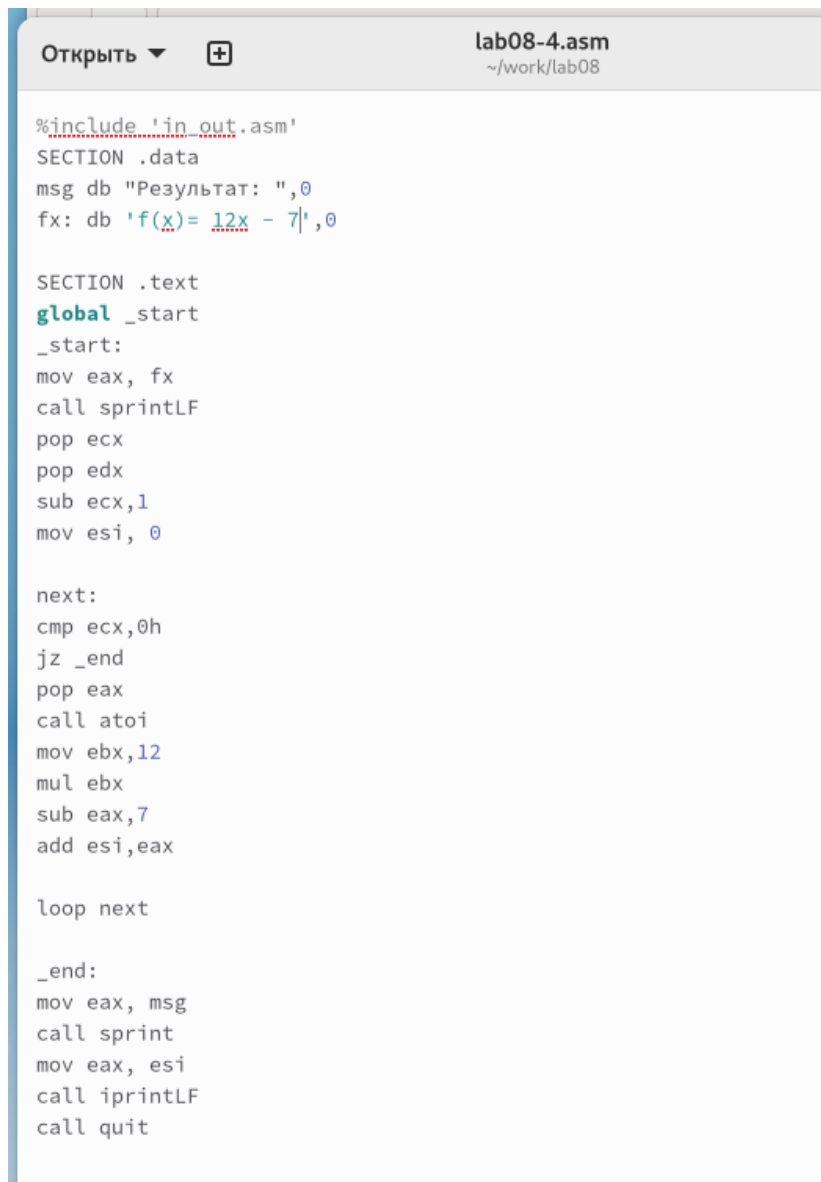

```
[malitvinov@fedora lab08]$  
[malitvinov@fedora lab08]$ nasm -f elf lab08-3.asm  
[malitvinov@fedora lab08]$ ld -m elf_i386 lab08-3.o -o lab08-3  
[malitvinov@fedora lab08]$ ./lab08-3  
Результат: 1  
[malitvinov@fedora lab08]$ ./lab08-3 3 4 5 6  
Результат: 360  
[malitvinov@fedora lab08]$
```

Рис. 2.12: Запуск программы

2.3 Задание для самостоятельной работы

Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x .

для варианта 13 $f(x) = 12x - 7$




```
Открыть ▾  lab08-4.asm  
~/work/lab08  
  
%include 'in_out.asm'  
SECTION .data  
msg db "Результат: ",0  
fx: db 'f(x)= 12x - 7|',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,12  
mul ebx  
sub eax,7  
add esi,eax  
  
loop next  
  
_end:  
mov eax, msg  
call sprint  
mov eax, esi  
call iprintLF  
call quit
```

Рис. 2.13: Изменение кода

```
[malitvinov@fedora lab08]$  
[malitvinov@fedora lab08]$ nasm -f elf lab08-4.asm  
[malitvinov@fedora lab08]$ ld -m elf_i386 lab08-4.o -o lab08-4  
[malitvinov@fedora lab08]$ ./lab08-4 1  
f(x)= 12x - 7  
Результат: 5  
[malitvinov@fedora lab08]$ ./lab08-4 1 3 4 5 6  
f(x)= 12x - 7  
Результат: 193  
[malitvinov@fedora lab08]$
```

Рис. 2.14: Запуск программы

3 Выводы

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