

# **Отчёт по лабораторной работе 8**

**Архитектура компьютера**

**Амарбаяр Чинхусэл**

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	20

## Список иллюстраций

2.1	Программа lab8-1.asm . . . . .	7
2.2	Запуск программы lab8-1.asm . . . . .	8
2.3	Программа lab8-1.asm . . . . .	9
2.4	Запуск программы lab8-1.asm . . . . .	10
2.5	Программа lab8-1.asm . . . . .	11
2.6	Запуск программы lab8-1.asm . . . . .	12
2.7	Программа lab8-2.asm . . . . .	13
2.8	Запуск программы lab8-2.asm . . . . .	14
2.9	Программа lab8-3.asm . . . . .	15
2.10	Запуск программы lab8-3.asm . . . . .	15
2.11	Программа lab8-3.asm . . . . .	16
2.12	Запуск программы lab8-3.asm . . . . .	17
2.13	Программа prog1.asm . . . . .	18
2.14	Запуск программы prog1.asm . . . . .	19

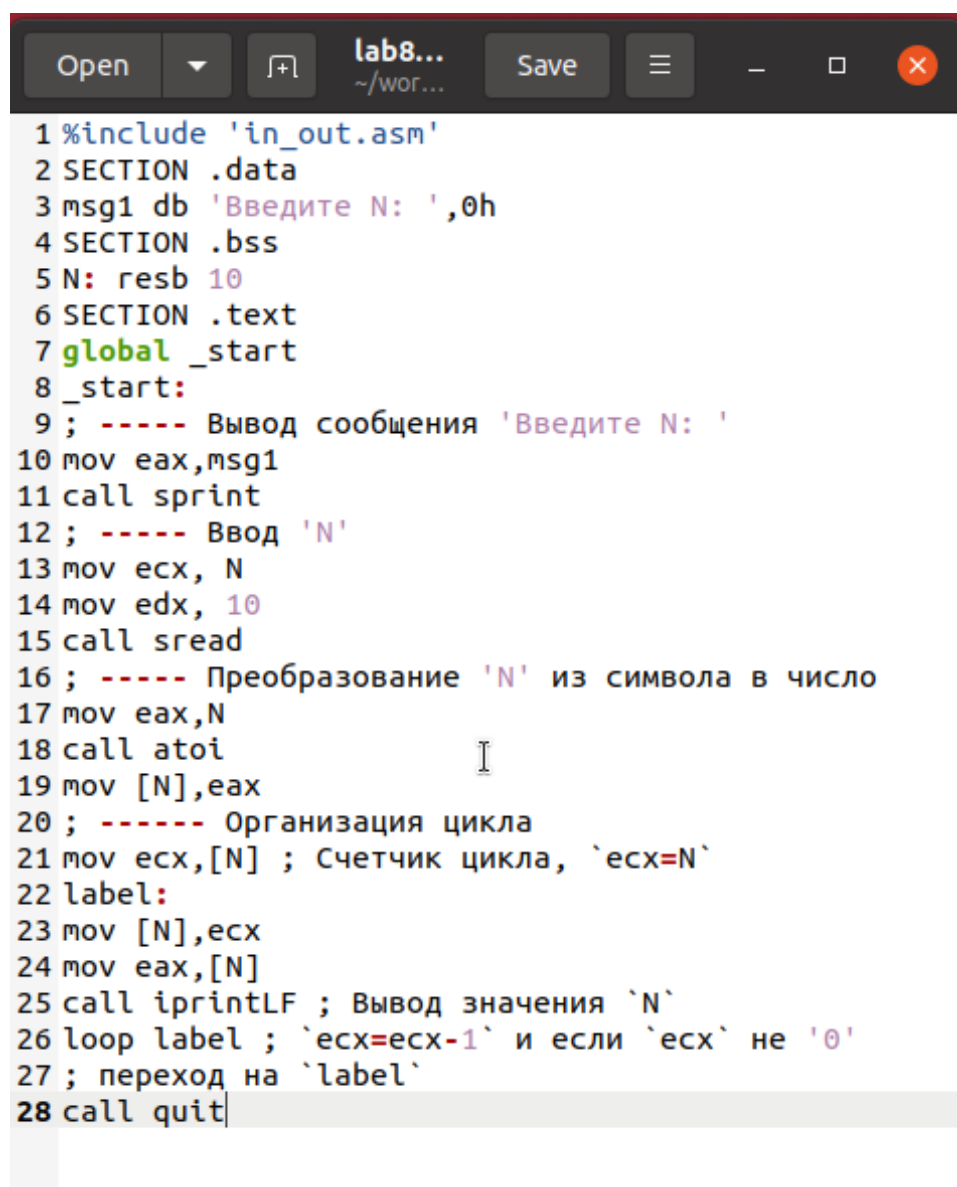
## **Список таблиц**

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

1. Я создал каталог для программ лабораторной работы № 8 и перешел в него. Затем я создал файл lab8-1.asm и написал в нем текст программы из листинга 8.1. После этого я создал исполняемый файл и проверил его работу.



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

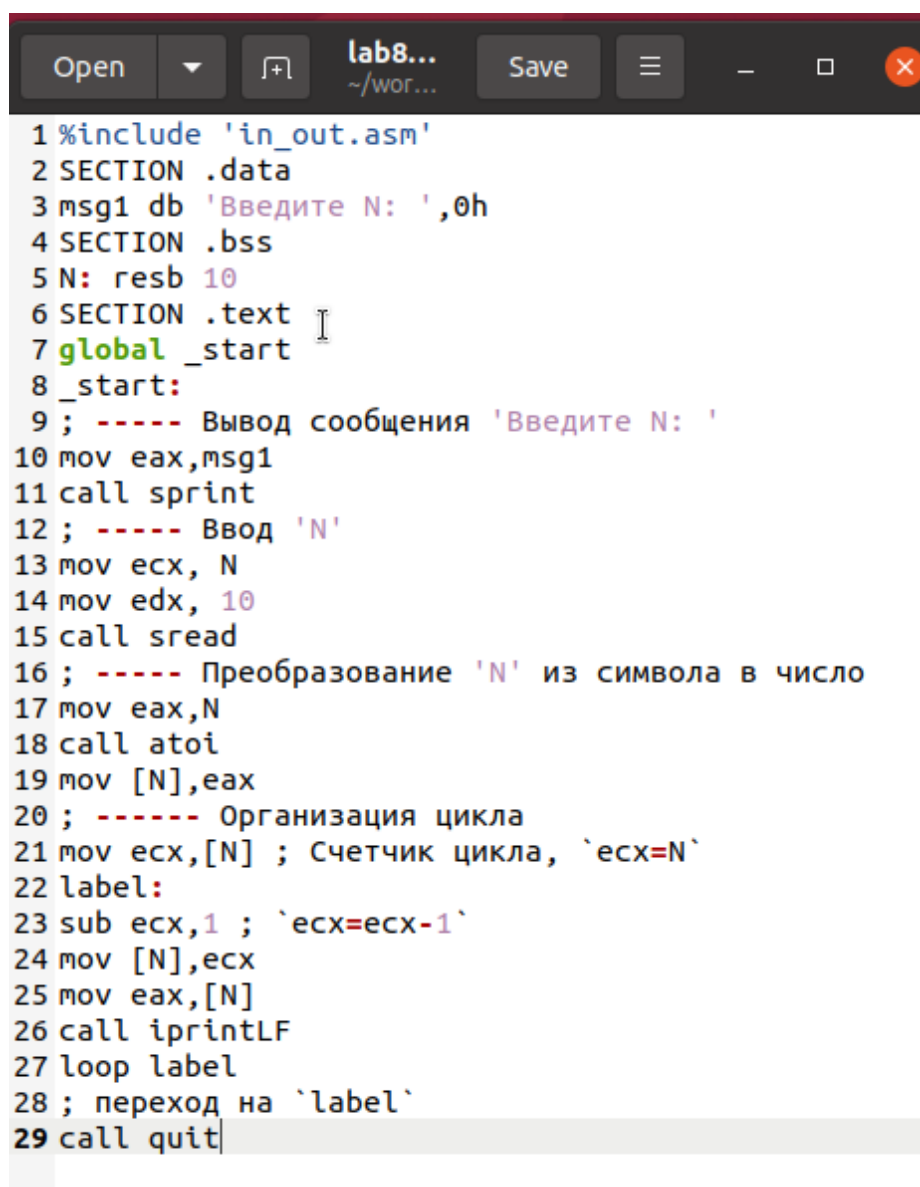
Рис. 2.1: Программа lab8-1.asm

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

Рис. 2.2: Запуск программы lab8-1.asm

2. Продолжил работу с файлом lab8-1.asm и внес изменения в текст программы, чтобы продемонстрировать проблему использования регистра esx в цикле loop, которая может привести к некорректной работе программы. Изменил значение регистра esx в цикле и затем снова создал исполняемый файл для проверки его работы. Если значение N было нечетным, программа входила в бесконечный цикл, а если значение N было четным, программа выводила только нечетные числа.





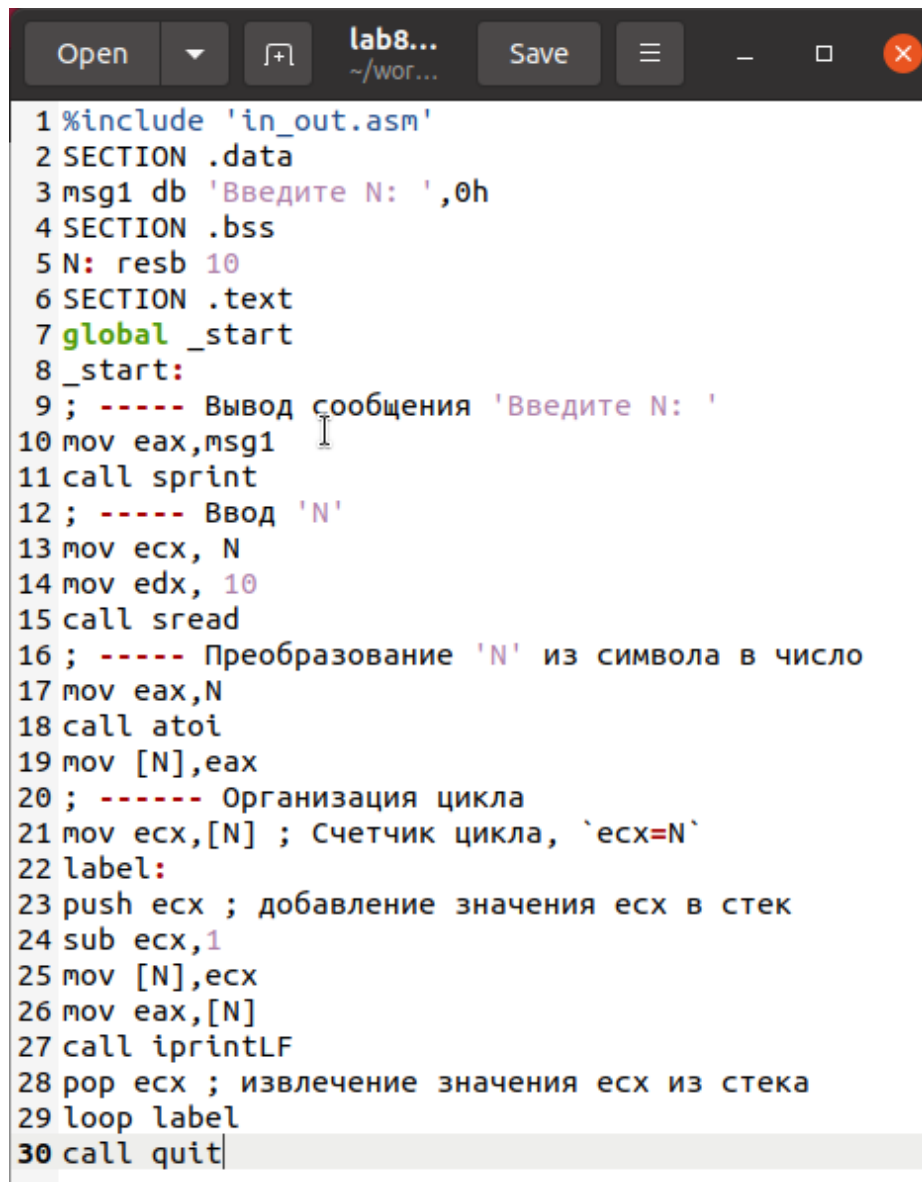
```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование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

Рис. 2.3: Программа lab8-1.asm

```
4294934748
4294934746
4294934744
4294934742
4294934740
4294934738
429^C
achinhusel@Ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
achinhusel@Ubuntu:~/work/arch-pc/lab08$
achinhusel@Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.4: Запуск программы lab8-1.asm

3. Чтобы сохранить корректность работы программы при использовании регистра `ecx` в цикле, нужно использовать инструкции стека. Внес изменения в текст программы, добавив команды `push` и `pop` для сохранения значения счетчика цикла `loop`. После этого я создал исполняемый файл и проверил его работу. Теперь программа выводила числа от  $N-1$  до 0, и количество проходов цикла соответствовало значению  $N$ .



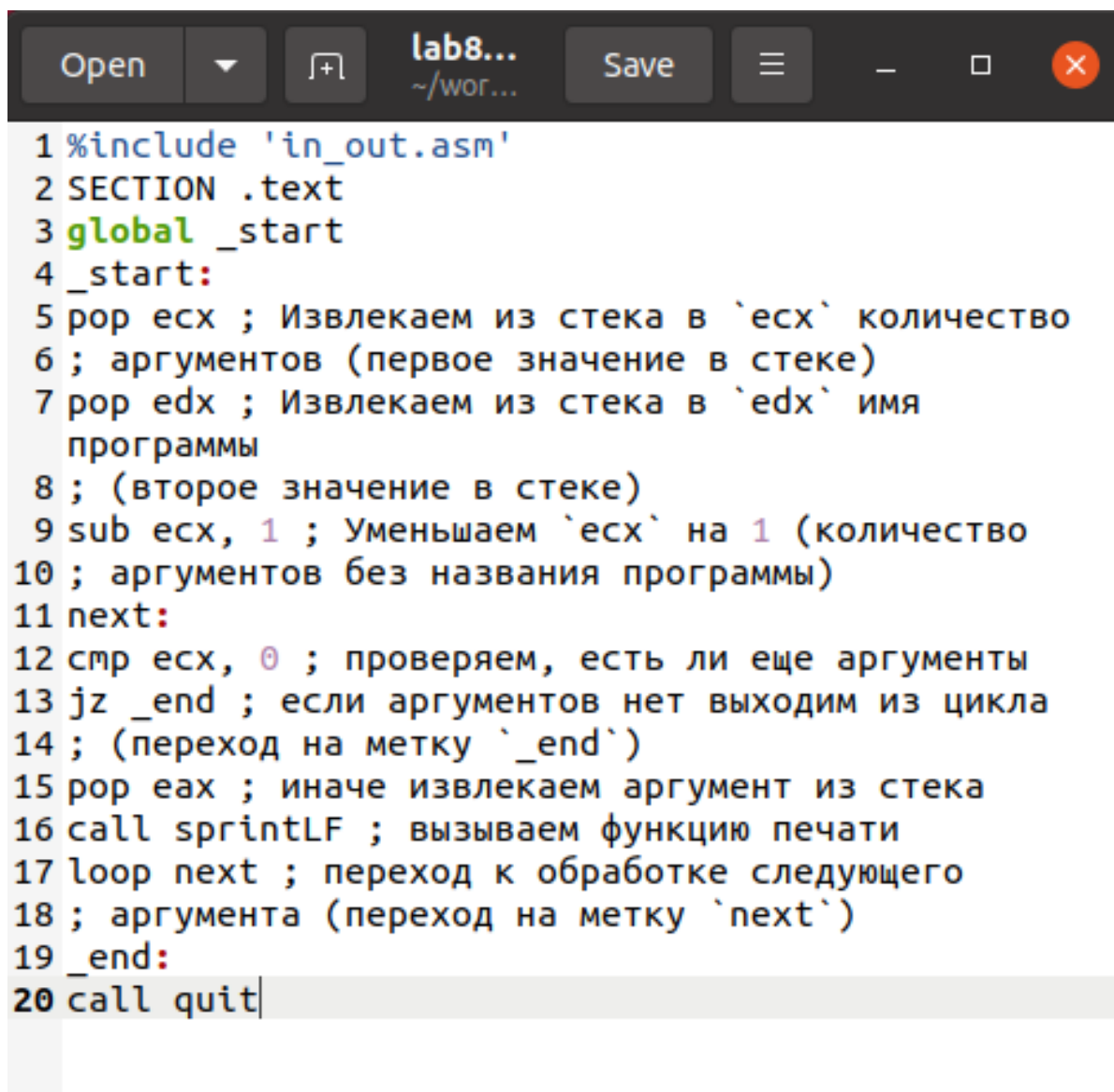
```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование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

Рис. 2.5: Программа lab8-1.asm

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

Рис. 2.6: Запуск программы lab8-1.asm

4. Я создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2. Затем создал исполняемый файл и запустил его, указав аргументы. Программа успешно обработала 5 аргументов.



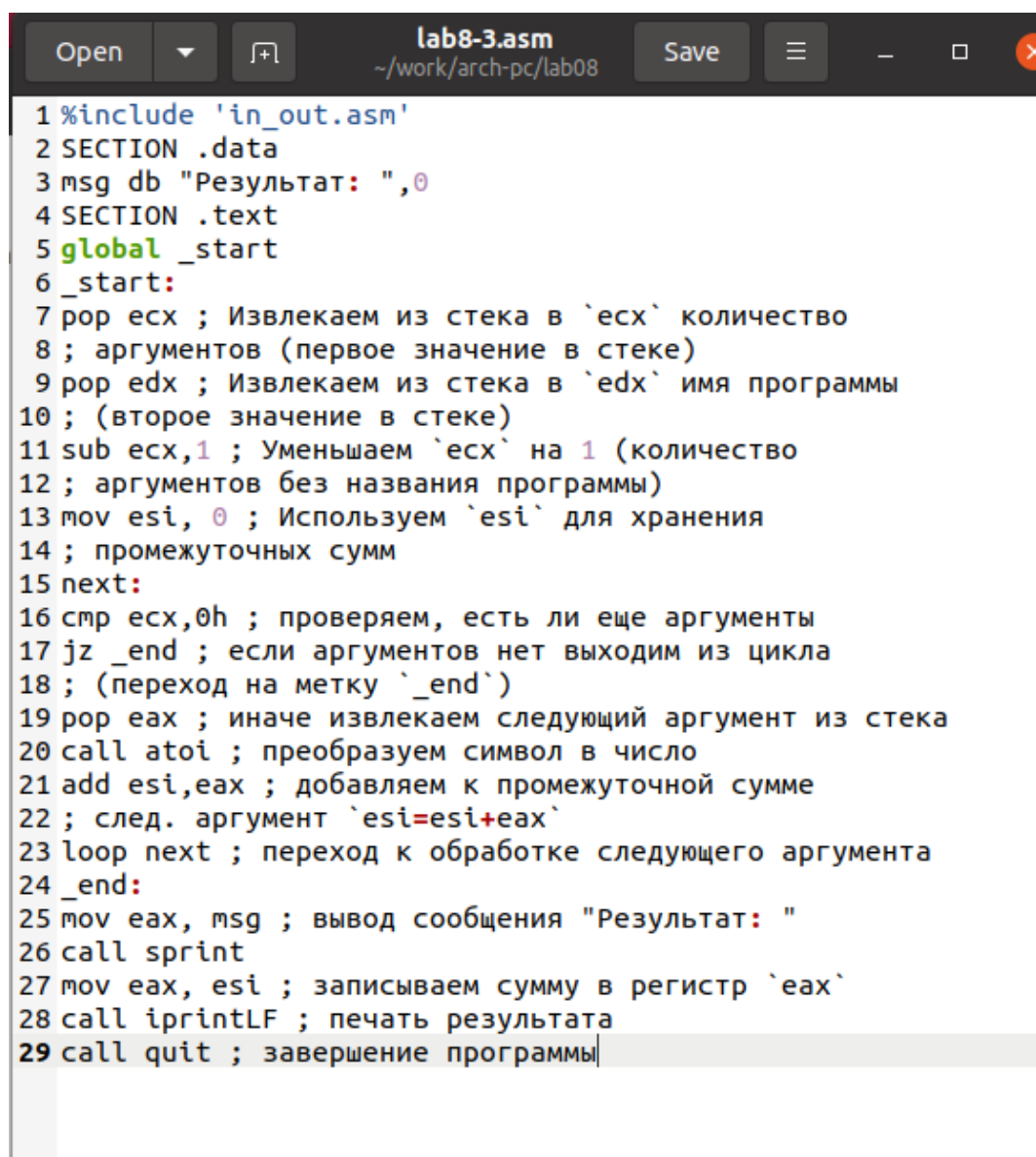
```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя
   программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14 ; (переход на метку `_end`)
15 pop eax ; иначе извлекаем аргумент из стека
16 call sprintf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

Рис. 2.7: Программа lab8-2.asm

```
achinhuse1@Ubuntu:~/work/arch-pc/lab08$  
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm  
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2  
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ./lab8-2 argument 1 argument 2 'argument  
3'  
argument  
1  
argument  
2  
argument 3  
achinhuse1@Ubuntu:~/work/arch-pc/lab08$  
achinhuse1@Ubuntu:~/work/arch-pc/lab08$
```

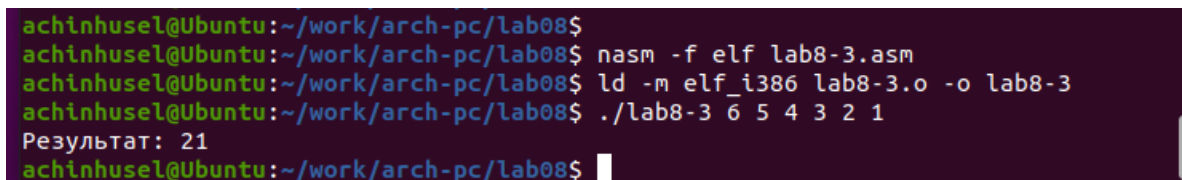
Рис. 2.8: Запуск программы lab8-2.asm

5. Рассмотрим еще один пример программы, которая выводит сумму чисел, переданных в программу как аргументы.



```
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
```

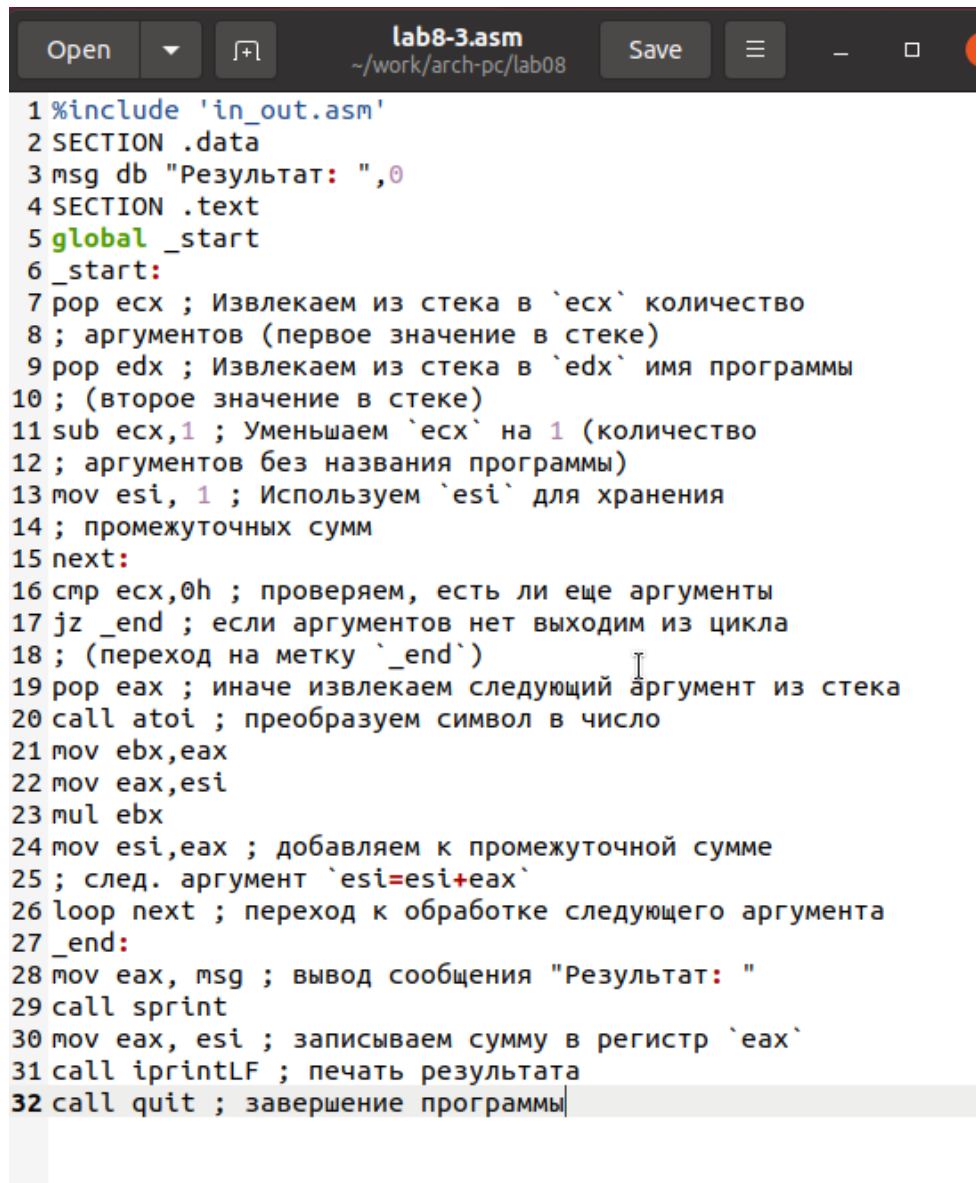
Рис. 2.9: Программа lab8-3.asm



```
achinhuse1@Ubuntu:~/work/arch-pc/lab08$
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 6 5 4 3 2 1
Результат: 21
achinhuse1@Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.10: Запуск программы lab8-3.asm

6. Внес изменения в текст программы из листинга 8.3 для вычисления произведения аргументов командной строки



```
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, 1 ; 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 mov ebx,eax
22 mov eax,esi
23 mul ebx
24 mov esi,eax ; добавляем к промежуточной сумме
25 ; след. аргумент `esi=esi+eax`
26 loop next ; переход к обработке следующего аргумента
27 _end:
28 mov eax, msg ; вывод сообщения "Результат: "
29 call sprint
30 mov eax, esi ; записываем сумму в регистр `eax`
31 call iprintLF ; печать результата
32 call quit ; завершение программы
```

Рис. 2.11: Программа lab8-3.asm



```

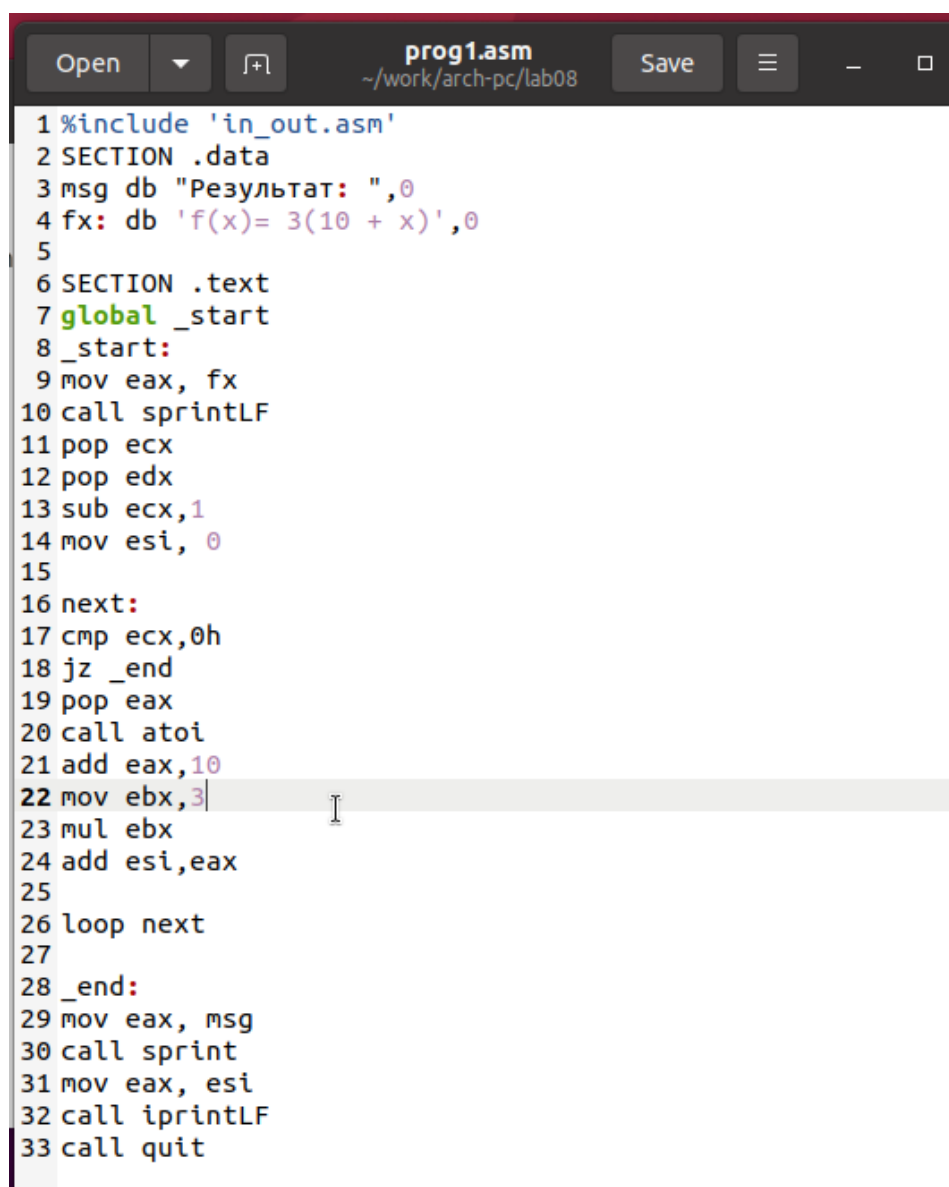
achinhuse1@Ubuntu:~/work/arch-pc/lab08$
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 6 5 4 3 2 1
Результат: 720
achinhuse1@Ubuntu:~/work/arch-pc/lab08$
achinhuse1@Ubuntu:~/work/arch-pc/lab08$

```

Рис. 2.12: Запуск программы lab8-3.asm

7. Написал 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. Создал исполняемый файл и проверил его работу на нескольких наборах  $x$ .

для варианта 20  $f(x) = 3(10 + x)$



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 3(10 + x)',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintf
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 pop eax
20 call atoi
21 add eax,10
22 mov ebx,3
23 mul ebx
24 add esi,eax
25
26 loop next
27
28 _end:
29 mov eax, msg
30 call sprintf
31 mov eax, esi
32 call iprintLF
33 call quit
```

Рис. 2.13: Программа prog1.asm

```
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ nasm -f elf prog1.asm
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 prog1.o -o prog1
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ./prog1
f(x)= 3(10 + x)
Результат: 0
achinhuse1@Ubuntu:~/work/arch-pc/lab08$
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ./prog1 0
f(x)= 3(10 + x)
Результат: 30
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ./prog1 1
f(x)= 3(10 + x)
Результат: 33
achinhuse1@Ubuntu:~/work/arch-pc/lab08$ ./prog1 6 5 4 3 21
f(x)= 3(10 + x)
Результат: 267
achinhuse1@Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.14: Запуск программы prog1.asm

## 3 Выводы

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